Sun Java™ System

Application Server Enterprise Edition 8.1 管理指南

2005Q1

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

文件号码 819-1553

目录

前言

本指南介绍如何配置和管理 Application Server。本前言包含有关以下主题的信息：

本书的读者

本管理指南适用于生产环境中的信息技术管理员。本指南假设您已经熟悉以下主题：

- 基本系统管理任务
- 安装软件
- 使用 Web 浏览器
- 启动数据库服务器
- 在终端窗口中发布命令

阅读本书之前

Application Server 是 Sun Java™ Enterprise System 的组件之一， Sun Java™ Enterprise System 是支持分布在网络或 Internet 环境中的企业应用程序的软件基础架构。您应该熟悉随 Sun Java Enterprise System 一起提供的文档，可以从 http://docs.sun.com/app/docs/prod/entsys.05q1 和 http://docs.sun.com/app/docs/prod/entsys.05q1?1=zh 位置联机访问该文档。

本书的结构

本指南的组织结构与管理控制台的布局相对应，管理控制台是用于管理 Application Server 的、基于浏览器的工具。每章都以概念性信息开头，随后的部分说明如何使用管理控制台执行特定的任务。

本书使用的约定

本节中的表格介绍了本书使用的约定。

印刷约定

下表介绍了本书对印刷约定的更改。

表 1 印刷约定

字样	含义	示例
AaBbCc123	API 和语言元素、HTML 标记、Web 站点 URL、命令名、文件名、目录路径名、计算机屏幕输出和样例代码。	编辑 `.login` 文件。 使用 `ls -a` 列出所有文件。 `% You have mail.`
AaBbCc123	您所键入的信息与计算机屏幕输出相对应时。	**`% su`** `Password:`
AaBbCc123	将用实际名称或值替换的命令或路径名中的占位符。	这些被称为 ***`class`*** 选项。 文件位于 *install-dir*`/bin` 目录中。
新词术语强调	新术语和要强调的文字。	请**勿**保存该文件。
《书名》	书名	阅读 《用户指南》的第 6 章。

符号

下表介绍了本书使用的符号约定。

表 2 符号约定

符号	说明	示例	含义
[ ]	包含可选的命令选项。	ls [-l]	无需 -l 选项。
{ | }	包含为所需命令选项提供的一组选择。	-d {y|n}	-d 选项需要您使用 y 参数或 n 参数。
-	连接需同时按下的多个击键。	Control-A	同时按 Control 键和 A 键。
+	连接需连续按下的多个击键。	Ctrl+A+N	按 Control 键，然后松开并依次按后面的键。
>	表示图形用户界面中的菜单项选定。	“文件” > “新建” > “模板”	从 “文件”菜单中，选择 “新建”。从 “新建”子菜单中，选择 “模板”。

默认路径和文件名

下表介绍了在本书中使用的默认路径和文件名。

表 3 默认路径和文件名

术语	说明
安装目录	默认情况下， Application Server 的安装目录位于以下位置： • Solaris™ 平台上的 Sun Java Enterprise System 的安装： /opt/SUNWappserver/appserver • Linux 平台上的 Sun Java Enterprise System 的安装： /opt/sun/appserver/ • 其他 Solaris 和 Linux 的安装 （非 root 用户）： 用户的主目录 /SUNWappserver • 其他 Solaris 和 Linux 的安装 （root 用户）： /opt/SUNWappserver • Windows 的所有安装： 系统驱动器 :\Sun\AppServer

表 3 默认路径和文件名 (续)

术语	说明
域根目录	默认情况下，包含所有域的目录位于以下位置： • Solaris 平台上的 Sun Java Enterprise System 的安装： /var/opt/SUNWappserver/domains/ • Linux 平台上的 Sun Java Enterprise System 的安装： /var/opt/sun/appserver/domains/ • 所有其他安装： 安装目录 /domains/
域目录	默认情况下，每个域目录都位于以下位置： 域根目录/域目录 在配置文件中，您可能会看到域目录显示为以下内容： ${com.sun.aas.instanceRoot}
实例目录	默认情况下，每个实例目录都位于以下位置： 域目录/实例目录

Shell 提示符

下表介绍了本书使用的 shell 提示符。

表 4 Shell 提示符

Shell	提示符
UNIX 或 Linux 上的 C shell	计算机名 %
UNIX 或 Linux 上的 C shell 超级用户	计算机名 #
UNIX 或 Linux 上的 Bourne shell 和 Korn shell	$
UNIX 或 Linux 上的 Bourne shell 和 Korn shell 超级用户	#
Windows 命令行	C:\

相关文档

http://docs.sun.com℠ Web 站点使您可以访问 Sun 的联机技术文档。您可以浏览归档文件或搜索特定书名或主题。

您可以在 *install_dir*/docs/index.htm 中找到有关官方规范的 URL 目录。此外，以下资源可能会有用。

常规 J2EE 信息：

《The J2EE 1.4 Tutorial》

http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

J2EE Blueprints：

http://java.sun.com/reference/blueprints/index.html

《Core J2EE Patterns: Best Practices and Design Strategies》由 Deepak Alur、John Crupi 和 Dan Malks 编写， Prentice Hall 出版

《Java Security》由 Scott Oaks 编写， O'Reilly 出版

使用 Servlet 和 JSP 文件编程：

《Java Servlet Programming》由 Jason Hunter 编写， O'Reilly 出版

《Java Threads, 2nd Edition》由 Scott Oaks 和 Henry Wong 编写， O'Reilly 出版

使用 EJB 组件编程：

《Enterprise JavaBeans》由 Richard Monson-Haefel 编写， O'Reilly 出版

使用 JDBC 编程：

《Database Programming with JDBC and Java》由 George Reese 编写， O'Reilly 出版

《JDBC Database Access With Java: A Tutorial and Annotated Reference (Java Series)》由 Graham Hamilton、 Rick Cattell 和 Maydene Fisher 编写

此文档集中包含的文档

Sun Java System Application Server 手册为联机文件，有可移植文档格式 (PDF) 和超文本标记语言 (HTML) 两种格式可供选择。

下表对包含在 Application Server 核心文档集中的文档进行了汇总。

表 5 此文档集中包含的文档

书名	说明
发行说明	软件和文档的最新信息。其中包括以表格形式对所支持的硬件、操作系统、 JDK 和 JDBC/RDBMS 所做的全面概述。
快速入门指南	如何开始使用 Sun Java System Application Server 产品。
Installation Guide	安装 Application Server 软件及其组件。
Deployment Planning Guide	评估系统需求和企业状况，确保以最适合您的站点的方式部署 Sun Java System Application Server。此外还介绍了部署应用服务器时应该注意的常见问题。
Developer's Guide	创建和实现要在 Application Server 上运行的 Java™ 2 Platform, Enterprise Edition（J2EE™ 平台）应用程序，这些应用程序遵循针对 J2EE 组件和 API 的开放式 Java 标准模型。其中包括有关开发者工具、安全性、汇编、部署、调试和创建生命周期模块的一般信息。
《 The J2EE 1.4 Tutorial 》	使用 J2EE 1.4 平台技术和 API 开发 J2EE 应用程序，并将这些应用程序部署到 Sun Java System Application Server。
管理指南	从管理控制台配置、管理和部署 Application Server 子系统和组件。
High Availability Administration Guide	配置和管理 Sun Java System Application Server 的高可用性功能。
Administration Reference	编辑 Sun Java System Application Server 的配置文件 `domain.xml`。
Upgrade and Migration Guide	将应用程序移植到新的 Sun Java System Application Server 编程模型，特别是从 Application Server 6.x 和 7 进行移植。该指南还介绍了可导致与产品规范不兼容的相邻产品版本和配置选项之间的差异。
Performance Tuning Guide	调优 Sun Java System Application Server 以提高性能。
Troubleshooting Guide	解决 Sun Java System Application Server 问题。
Error Message Reference	解决 Sun Java System Application Server 错误消息。
Reference Manual	可用于 Sun Java System Application Server 的实用程序命令，以手册页样式编写。其中包括 `asadmin` 命令行界面。

其他服务器文档

有关其他服务器文档，请转至以下位置：

- Message Queue 文档
  http://docs.sun.com/db?p=prod/s1.s1msgqu
- Directory Server 文档
  http://docs.sun.com/coll/DirectoryServer_04q2
- Web Server 文档
  http://docs.sun.com/coll/S1_websvr61_en

访问 Sun 联机资源

有关产品下载、专业服务、修补程序及支持和其他开发者信息，请转至以下位置：

- 下载中心
  http://wwws.sun.com/software/download/
- 专业服务
  http://www.sun.com/service/sunps/sunone/index.html
- Sun 企业服务、 Solaris 修补程序和支持
  http://sunsolve.sun.com/
- 开发者信息
  http://developers.sun.com/prodtech/index.html

与 Sun 技术支持部门联系

如果您有关于此产品的技术问题，并且在此产品文档中找不到其答案，请访问 http://www.sun.com/service/contacting。

相关的第三方 Web 站点引用

Sun 对本文档中提到的第三方 Web 站点的可用性不承担任何责任。对于此类站点或资源中的 （或通过它们获得的）任何内容、广告、产品或其他材料， Sun 并不表示认可，也不承担任何责任。对于因使用或依靠此类站点或资源中的 （或通过它们获得的）任何内容、产品或服务而造成的、宣称的或连带产生的实际或名义损坏或损失， Sun 概不负责，也不承担任何责任。

Sun 欢迎您提出意见

Sun 致力于提高其文档质量，并十分乐意收到您的意见和建议。

为了分享您的意见和建议，请访问 http://docs.sun.com 并单击 "Send Comments"。在联机表单中，提供了文档标题和文件号码。文件号码是一个七位或九位的数字，可以在书的标题页或文档的顶部找到。例如，本书的标题是 《Sun Java System Application Server 2005Q1 管理指南》，文件号码是 819-1553。在您提出意见时，可能需要在表单中输入英文版书名和文件号码，本书的英文版文件号码和书名为：819-0215， Sun Java System Application Server 2005Q1 Administration Guide。

第 1 章

入门

本章介绍 Sun Java™ System Application Server 以及其基本的管理任务。本章包含以下几节：

关于 Sun Java System Application Server

什么是 Application Server？

Application Server 为开发、部署和管理企业应用程序提供了强大的 J2EE 平台。主要功能包含事务管理、性能、可伸缩性、安全性和集成。Application Server 支持从 Web 发布到企业规模事务处理等多项服务，同时使开发者能够建立基于 Java Server Page (JSP™)、 Java Servlet 和企业 JavaBean™ (EJB™) 技术的应用程序。

Application Server Enterprise Edition 提供了高级群集和故障转移技术。这些功能可以帮助您运行可扩展的且具有高可用性的 J2EE 应用程序。

- **群集**－群集是一组应用程序服务器实例，它们作为一个逻辑实体一起工作。群集中的每个 Application Server 实例都拥有相同的配置，并被部署了相同的应用程序。

  通过将 Application Server 实例添加到群集从而增加系统容量，便实现了水平伸缩。可以在不中断服务的情况下将 Application Server 实例添加到群集。HTTP、 RMI/IIOP 和 JMS 负载平衡系统会将请求分布到群集中正常运行的 Application Server 实例中。

- **高可用性**－可用性允许对群集中的 Application Server 实例进行故障转移保护。一个 Application Server 实例出现故障时，其他 Application Server 实例将接管指定给该故障服务器的会话。会话信息存储在高可用性数据库 (HADB) 中。HADB 支持 HTTP 会话和有状态会话 Bean 的持久性。

Application Server 体系结构

本节介绍了图 1-1，此图显示了 Application Server 的高级别体系结构。

图 1-1 Application Server 体系结构

- **容器**－容器是一种运行时环境，它为 J2EE 组件提供安全性和事务管理等服务。图 1-1 显示了两种类型的 J2EE 容器：Web 和 EJB。 Web 组件 （如 JSP 页面和 Servlet）在 Web 容器内运行。企业 Bean （EJB 技术的组件）在 EJB 容器内运行。
- **客户机访问**－运行时，浏览器客户机通过 HTTP （在 Internet 中使用的协议）与 Web 服务器进行通信来访问 Web 应用程序。 HTTPS 协议用于需要安全通信的应用程序。企业 Bean 客户机通过 IIOP 协议或 IIOP/SSL （安全）协议与对象请求代理 (ORB) 进行通信。 Application Server 具有分别用于 HTTP 协议、HTTPS 协议、 IIOP 协议和 IIOP/SSL 协议的侦听器。每个侦听器独占使用特定的端口号。
- **Web 服务**－在 J2EE 平台上，可以部署一个 Web 应用程序，该应用程序可以提供由用于基于 XML 的 RPC 的 Java API (JAX-RPC) 实现的 Web 服务。 J2EE 应用程序或组件还可以是其他 Web 服务的客户机。应用程序通过用于 XML 注册表的 Java API (JAXR) 访问 XML 注册表。
- **用于应用程序的服务**－ J2EE 平台旨在使容器为应用程序提供服务。图 1-1 显示了以下服务：
  - 命名－命名和目录服务将对象绑定到名称。 J2EE 应用程序通过查找对象的 JNDI 名称来找到对象。 JNDI 表示 Java 命名和目录接口 API。
  - 安全性－ Java 容器授权合同 (JACC) 是一组为 J2EE 容器定义的安全性合同。根据客户机的标识，容器限制对容器的资源和服务的访问。
- **事务管理**－事务是不可分的工作单元。例如，在银行帐户之间转帐是一个事务。事务管理服务用于确保完全完成事务或将事务回滚。

访问外部系统

J2EE 平台使应用程序能够访问应用程序服务器之外的系统。应用程序通过称为资源的对象连接到这些系统。管理员的职责之一是资源配置。 J2EE 平台使得可以通过以下 API 和组件访问外部系统：

- **JDBC** －数据库管理系统 (DBMS) 提供了用于存储、组织和检索数据的工具。大多数商业应用程序将数据存储在关系数据库中，这些应用程序通过 JDBC API 访问关系数据库。由于数据库中的信息保存在磁盘上并在应用程序结束之后仍然存在，因此通常将数据库中的信息称为持久性信息。 Application Server 捆绑包含 PointBase DBMS。
- **消息传送**－消息传送是软件组件或应用程序之间的一种通信方法。消息传送客户机可以向任何其他客户机发送消息，也可以从任何其他客户机接收消息。应用程序通过 Java 消息传送服务 (JMS) API 访问消息传送提供者。 Application Server 包含一个 JMS 提供者。

- **连接器**－J2EE 连接器体系结构允许 J2EE 应用程序和现有企业信息系统 (EIS) 之间的集成。应用程序通过称为连接器或资源适配器的可移植 J2EE 组件访问 EIS。
- **JavaMail** －应用程序通过 JavaMail API 连接到 SMTP 服务器以发送和接收电子邮件。
- **服务器管理**－图 1-1 的右下角显示了由 Application Server 的管理员执行的一些任务。例如，管理员部署 （安装）应用程序并监视服务器的性能。这些任务通过 Application Server 提供的管理工具来执行。
- 管理工具
- Application Server 包括三种管理工具：
  - 管理控制台
  - asadmin 实用程序
  - Application Server Management Extension (AMX)

管理控制台

管理控制台是一种基于浏览器的工具，具有易于浏览的界面和联机帮助。本手册提供了使用管理控制台的逐步说明。必须运行管理服务器才能使用管理控制台。

安装 Application Server 时，您为服务器选择了端口号，或使用的是默认端口号 4849。还指定了用户名和主密码。

要启动管理控制台，请在 Web 浏览器中键入以下内容：

```
https://hostname:port
```

例如：

```
https://kindness.sun.com:4949
```

如果管理控制台在安装了 Application Server 的计算机上运行，请将 `localhost` 指定为主机名。

在 Windows 上，从 “开始”菜单启动 Application Server 管理控制台。

安装程序将创建默认管理域 （名为 `domain1`），并使用默认端口号 4849，还会创建一个独立于域管理服务器 (DAS) 的实例。安装之后，还可以创建其他管理域。每个域都有自己的域管理服务器，该服务器具有唯一的端口号。为管理控制台指定 URL 时，请确保使用要管理的域的端口号。

如果配置中包括远程服务器实例，请创建节点代理以便管理和简化远程服务器实例。节点代理负责创建、启动、停止和删除服务器实例。使用命令行界面 (CLI) 命令可以设置节点代理。

asadmin 实用程序

asadmin 实用程序是一个命令行工具。使用 asadmin 实用程序和与其关联的命令可以执行在管理控制台中可执行的相同任务集。例如，启动和停止域、配置服务器以及部署应用程序。

可以在 Shell 的命令提示符下使用这些命令，也可以从其他脚本和程序调用这些命令。使用这些命令可以自动执行重复的管理任务。

要启动 asadmin 实用程序，请使用：

```
$ asadmin
```

要列出 asadmin 中的可用命令，请使用：

```
asadmin> help
```

也可以在 Shell 的命令提示符下发出 asadmin 命令：

```
$ asadmin help
```

要查看命令的语法和示例，请在命令名称后面键入 help。例如：

```
asadmin> help create-jdbc-resource
```

给定命令的 asadmin help 信息将显示此命令的 Unix 手册页。也可以以 HTML 格式查看这些手册页。

Application Server Management Extension (AMX)

Sun Java System **A**pplication Server **M**anagement e**X**tension 是一个 API，它显示 Application Server 的所有配置，并将 JMX 管理 Bean 当作实现 AMX 接口的、易于使用的客户端动态代理来进行监视。

有关使用 Application Server Management Extension 的更多信息，请参见 《Sun Java System Application Server Developer's Guide》中的 "Using the Java Management Extensions (JMX) API" 一章。

Application Server 配置

配置 Application Server

Application Server 域是为帮助管理员管理系统配置而创建的逻辑或物理单元。一个域被分为很多包括实例和节点代理的较小单元。服务器实例是在一个物理计算机上运行 Application Server 的一个 Java 虚拟机 (JVM)。每个域都有一个或多个实例。域还必须至少有一个相关联的节点代理才能使实例正常工作。可以将域组合在一起以创建一个群集。群集使管理员可以管理多组硬件和软件。

配置域

管理域提供了基本的安全性结构，凭借此结构，不同的管理员可以管理应用程序服务器实例的特定组 （域）。通过将服务器实例分组到单独的域中，不同的组织和管理员可以共享一个 Application Server 安装。每个域都有自己的独立于其他域的配置、日志文件和应用程序部署区域。如果更改某个域的配置，其他域的配置不会受到影响。

每个管理控制台会话均允许您配置和管理域。如果创建了多个域，则必须启动另外一个管理控制台会话以管理每个域。每个域都有自己的域管理服务器 (DAS)，该服务器具有唯一的端口号。每个管理域可以有多个应用程序服务器实例。但是，一个应用程序服务器实例只能属于一个域。安装 Application Server 时，将自动创建名为 `domain1` 的管理域。

创建域

域是使用 create-domain 命令创建的。以下示例命令将创建名为 mydomain 的域。管理服务器在端口 1234 上进行侦听，管理用户名为 hanan。该命令提示输入管理密码和主密码。

```
$ asadmin create-domain --adminport 80 --adminuser hanan mydomain
```

要为 mydomain 域启动管理控制台，请输入以下 URL：

http://*hostname*:80

对于前面的 create-domain 示例，域的日志文件、配置文件和部署的应用程序现在位于以下目录中：

install_dir/domains/mydomain

要在其他位置创建域的目录，请指定 --domaindir 选项。要查看完整的命令语法，请键入 asadmin help create-domain。

删除域

使用 asadmin delete-domain 命令可以删除域。只有可以管理域的操作系统用户 （即超级用户）可以成功地执行此命令。例如，要删除名为 mydomain 的域，请键入以下命令：

```
$ asadmin delete-domain mydomain
```

列出域

使用 asadmin list-domains 命令可以找到在计算机上创建的域。要列出默认的 *install_dir*/domains 目录中的域，请键入以下命令：

```
$ asadmin list-domains
```

要列出在其他目录中创建的域，请指定 --domaindir 选项。

启动域

启动域时，将启动管理服务器和应用程序服务器实例。启动应用程序服务器实例之后，应用程序服务器实例将持续运行、侦听并接受请求。必须单独启动各个域。

要启动域，请键入 `asadmin start-domain` 命令并指定域名。例如，要启动默认域 (`domain1`)，请键入以下内容：

```
$ asadmin start-domain domain1
```

如果只有一个域，则可以省略域名。要查看完整的命令语法，请键入 `asadmin help start-domain`。如果省略了密码数据，系统将提示您提供此数据。

要在 Windows 中启动默认域，请：

在 Windows “开始”菜单中，依次选择 “程序” -> "Sun Microsystems" -> "Application Server" -> “启动管理服务器”。

重新启动服务器或域

重新启动服务器与重新启动域相同。要重新启动域或服务器，请停止然后再启动域。

停止域

停止域将关闭该域的管理服务器和应用程序服务器实例。停止域时，服务器实例将停止接收新的连接，然后等待所有未完成的连接完成。由于服务器实例必须完成其关闭进程，因此该进程需要几秒钟时间。停止域时，管理控制台或大多数 `asadmin` 命令都无法使用。

要停止域，请键入 `asadmin stop-domain` 命令并指定域名。例如，要停止默认域 (`domain1`)，请键入以下内容：

```
$ asadmin stop-domain domain1
```

如果只有一个域，则域名是可选键入项。要查看完整的语法，请键入 `asadmin help stop-domain`。

要使用管理控制台停止域，请执行以下步骤：

- 在树组件中，选择 “独立实例”节点下的服务器 （管理服务器）。
- 在 “常规信息”页面中，单击 “停止服务器”。

要在 Windows 中停止默认域，请：

在 “开始”菜单中，依次选择 “程序” -> "Sun Microsystems" - > "Applicationserver" -> “停止管理服务器”。

重新创建域管理服务器

要进行镜像和提供域管理服务器 (DAS) 的工作副本，您必须拥有以下设备：

- 一台包含原始 DAS 的计算机 (machine1)。
- 一台包含群集的计算机 (machine2)，该群集具有运行应用程序并满足客户机需要的服务器实例。该群集是使用第一台计算机上的 DAS 配置的。
- 一台备份计算机 (machine3)，当第一台计算机崩溃时，需要在该备份计算机上重新创建 DAS。

注 必须对第一台计算机上的 DAS 进行备份。请使用 `asadmin backup-domain` 来备份当前域。

DAS 移植的步骤

以下步骤用于将域管理服务器从第一台计算机 (machine1) 移植到第三台计算机 (machine3)：

1. 设置第三台计算机，方法是：在第三台计算机上安装与在第一台计算机上安装的应用程序服务器相同的应用程序服务器。

   为了可以在第三台计算机上正确地恢复 DAS 并且不会发生路径冲突，您必须执行此操作。

   - 使用命令行 （交互式）模式来安装应用程序服务器管理软件包。要激活交互式命令行模式，请使用 `console` 选项调用安装程序：

     ***./ bundle_filename* `-console`**

     要使用命令行界面进行安装，您必须具有超级用户权限。

   - 要安装默认域，请取消选择该选项。

     只有具有相同体系结构并具有**完全**相同的安装路径 （即使用相同的 *install_dir*）的两台计算机才支持备份域的恢复。

2. 将第一台计算机上的备份 ZIP 文件复制到第三台计算机上的 *install_dir*/domains 目录中。也可以通过 FTP 方式传输文件。

3. 执行 asadmin restore-domain 命令以将 ZIP 文件恢复到第三台计算机：

   ```
   asadmin restore-domain --filename
   install_dir/domains/sjsas_backup_v00001.zip domain1
   ```

   可以备份任何域。但是，在重新创建域时，域名称应与原始域名称相同。

4. 将第三台计算机上的 *install_dir*/domains/domain1/generated/tmp 目录的权限更改为与第一台计算机上相同目录的权限相匹配。

   该目录的默认权限为：?drwx------? （或 700）。

   例如：

   ```
   chmod 700 install_root/domains/domain1/generated/tmp
   ```

   以上示例假定您备份的是 *domain1*。如果备份的是其他名称的域，应使用要备份的域的名称替换上面的 *domain1*。

5. 更改第三台计算机的 domain.xml 中的主机属性值：

6. 更新第三台计算机上的 *install_root*/domains/*domain1*/config/domain.xml。

   例如：

   搜索 *machine1* 并将其替换为 *machine3*。这样，您就可以将：

   ```
   <jmx-connector><property name=client-hostname value=machine1/>...
   ```

   更改为：

   ```
   <jmx-connector><property name=client-hostname value=machine3/>...
   ```

7. 将：

   ```
   <jms-service... host=machine1.../>
   ```

   更改为：

   ```
   <jms-service... host=machine3.../>
   ```

8. 在 machine3 上启动已恢复的域：

   ```
   asadmin start-domain --user admin_user --password admin_password
   domain1
   ```

9. 在 machine2 上更改节点代理下的 DAS 主机属性值：

10. 更改 machine2 中的 *install_dir*/nodeagents/*nodeagent*/agent/config/das.properties 中的 agent.das.host 属性值。
11. 在 machine2 上重新启动节点代理。

注 使用 asadmin start-instance 命令启动群集实例，以使这些实例与已恢复的域同步。

Application Server 实例

关于 Application Server 实例

Sun Java System Application Server 在安装时将创建一个称为 server 的应用程序服务器实例。如果需要，可以删除此服务器实例，并创建一个其他名称的新实例。

每个 Sun Java System Application Server 实例都有自己的 J2EE 配置、 J2EE 资源、应用程序部署区域和服务器配置设置。因此对一个应用程序服务器实例所做的更改不会影响其他应用程序服务器实例。一个管理域中可以包含多个应用程序服务器实例。

对于很多用户来说，一个应用程序服务器实例即可满足他们的需要。但是，根据用户环境的不同，可能需要创建一个或多个附加应用程序服务器实例。例如，在开发环境中，可以使用不同的应用程序服务器实例测试不同的 Sun Java System Application Server 配置，或者比较和测试不同的应用程序部署。由于添加或删除应用程序服务器实例比较容易，因此在开发过程中可以使用这些实例创建临时 “沙盒”区来进行测试。

此外，还可以为每个应用程序服务器实例创建虚拟服务器。在一个已安装的应用程序服务器实例中，您可以提供公司或个人域名、 IP 地址和某些管理功能。对于用户，就好比他们拥有自己的 Web 服务器，但无需进行硬件和基础服务器的维护。这些虚拟服务器不能跨应用程序服务器实例使用。有关虚拟服务器的更多信息，请参见配置 JVM 常规设置。

在操作部署中，很多情况下都可以使用虚拟服务器代替多个应用程序服务器实例。但是，如果虚拟服务器不能满足需求，您也可以使用多个应用程序服务器实例。

Sun Java System Application Server 实例不会自动启动。启动某个实例后，该实例将一直运行，直至您将其停止。停止应用程序服务器实例后，它将不再接受新的连接，只是等待未完成的连接完成。如果您的计算机崩溃或脱机，则服务器将退出，并且正在处理的所有请求都将丢失。

定义 Application Server 实例

应用程序服务器实例构成了应用程序部署的基础。每个实例均属于一个域，并有自己的目录结构、配置和已部署的应用程序。每个服务器实例还包含了 J2EE 平台的 Web 和 EJB 容器。每个新的服务器实例必须包含对节点代理名称的引用，该名称定义实例将要驻留的计算机。

可以创建三种类型的服务器实例。每个服务器实例只能是其中的一种类型：

- 在**独立服务器**实例中，其他任何服务器实例或群集不能共享其配置。
- 在**共享服务器实例**中，其他实例或群集可以共享其配置。
- 在**群集服务器实例**中，群集中的其他实例可以共享其配置。

  图 1-2 详细显示了一个应用程序服务器实例。应用服务器实例是 Application Server 企业版的群集、负载平衡和会话持久性功能中的构成块。

- **定义和使用群集**－群集是共享一套相同的应用程序、资源和配置信息的一组服务器实例。服务器实例只能属于一个群集。特别值得一提的是，使用群集可以通过在多台计算机上分布负载来促进负载平衡，并可以通过实例级别的故障转移来提供高可用性。

图 1-2 Sun Java System Application Server 实例

关于独立实例

Sun Java System Application Server 实例不会自动启动。启动某个实例后，该实例将一直运行，直至您将其停止。停止应用程序服务器实例后，它将不再接受新的连接，只是等待未完成的连接完成。如果您的计算机崩溃或脱机，则服务器将退出，并且正在处理的所有请求都将丢失。

查看常规服务器信息

通过 “常规”选项卡可以执行以下任务：

- 单击 “启动实例”以启动实例。
- 单击 “停止实例”以停止实例。
- 单击 “查看日志文件”，以打开服务器日志查看器。
- 单击 “轮转日志文件”，以轮转实例的日志文件。该操作将安排日志文件以进行轮转。实际的轮转将在下一次向日志文件写入条目时发生。默认服务器 (DAS) 的轮转将立即发生，但其他独立服务器的轮转将延迟。
- 单击 “JNDI 浏览”以浏览正在运行的实例的 JNDI 树。
- 单击 “恢复事务”以恢复未完成的事务。

此外，您可以选择以下选项卡以执行其他任务：

- “应用程序”选项卡：部署选定的应用程序。
- “资源”选项卡：管理选定的资源。
- “属性”选项卡：配置实例特定的属性。
- “监视”选项卡：查看 JVM、服务器、线程池、 HTTP 服务和事务服务的监视数据。
- “高级”选项卡：设置用于部署应用程序的常规属性。

管理应用程序

通过 “应用程序”选项卡，您可以启用、禁用和部署与实例相关联的选定应用程序。

要部署应用程序，请执行以下步骤：

1. 选中所需应用程序的复选框。
2. 从 “部署”下拉式菜单中，选择要部署的应用程序模块的类型：
   - 企业应用程序：位于 EAR （企业应用程序归档）文件或目录中的 J2EE 应用程序。
   - Web 应用程序：封装在 WAR （Web 应用程序归档）文件或目录中的 Web 资源 （例如 JavaServer Pages [JSP]、 servlet 和 HTML 页面）的集合，。
   - EJB 模块：包含在 EJB JAR （Java 归档）文件或目录中的一个或多个 Enterprise JavaBeans (EJB)。
   - 连接器模块：连接至企业信息系统 (EIS) 并封装在 RAR （资源适配器归档）文件或目录中。
   - 生命周期模块：在被服务器生命周期中的一个或多个事件触发时执行任务。
   - 应用程序客户机模块：也称为 J2EE 应用程序客户机 JAR 文件，它包含客户机的服务器端例程。

管理资源

通过 “资源”选项卡，您可以启用、禁用和创建与实例相关联的新资源类型。

要创建新的资源类型，请执行以下步骤：

1. 选中所需资源的复选框。
2. 从 “新建”下拉式菜单中，选择要创建并与该实例相关联的资源类型：
   - JDBC：为应用程序提供连接数据库的方法。
   - 持久性管理器：具有容器管理的持久性 Bean （用于向下兼容性）的应用程序需要持久性管理器。
   - JMS 连接工厂：允许应用程序以编程方式创建其他 JMS 对象的对象。
   - JMS 目标：表示 JavaMail API 中的邮件会话， JavaMail API 提供了一个独立于平台和协议的框架来建立邮件和消息传送应用程序。

- JavaMail：提供了一个独立于平台和协议的框架来建立邮件和消息传送应用程序。
- 自定义：表示具有已定义的 JNDI 子上下文、资源类型和工厂类的非标准资源。
- 外部：使应用程序能够查找位于轻量目录访问协议 (LDAP) 系统信息库中的外部资源对象。
- 连接器：程序对象，为应用程序提供到企业信息系统 (EIS) 的连接。
- 管理对象：配置 JSR-160 兼容的远程 JMX 连接器

Administration Server 高级设置

Administration Server 高级设置使您可以设置用于部署应用程序的常规属性。使用这些属性，可以确保和监视对部署的应用程序所做的更改进行检测，以及重新装入已修改的类。

设置应用程序配置

如果启用了动态重新装入，服务器将定期检查已部署的应用程序的文件中的更改并按照更改自动重新装入应用程序。动态重新装入在开发环境中很有用，因为它允许快速测试代码更改。但在生产环境中，动态重新装入可能会使性能降低。

动态重新装入适用于开发环境。它与会话持久性 （一种生产环境功能）不兼容。如果启用了动态部署，请勿启用会话持久性。

注 动态重新装入仅适用于默认服务器实例。

要在 “应用程序配置”页面中配置动态重新装入，请配置以下项目：

- 重新装入：使用 “已启用”复选框来启用或禁用动态重新装入。
- 重新装入轮询时间间隔：指定服务器检查已部署的应用程序中的更改的频率。
- 管理会话超时：指定管理会话超时且必须重新登录之前的时间。

设置自动部署

自动部署功能使您能够将预封装的应用程序或模块复制到 *install_dir*/domains/*domain_dir*/autodeploy 目录来部署该应用程序或模块。

例如，将名为 hello.war 的文件复制到 *install_dir*/domains/domain1/autodeploy 目录。要取消部署应用程序，请从 autodeploy 目录中删除 hello.war 文件。

自动部署功能适用于开发环境。它与会话持久性 （一种生产环境功能）不兼容。如果启用了动态部署，请勿启用会话持久性。

注 自动部署仅适用于默认服务器实例。

要在 “应用程序配置”页面中配置自动部署设置，请执行以下步骤：

1. 通过选中或取消选中 “已启用”复选框来启用或禁用自动部署。
2. 在 “自动部署轮询时间间隔”字段中，指定服务器检查自动部署目录中的应用程序文件或模块文件的频率。更改轮询时间间隔不会影响部署应用程序或模块所需的时间。
3. 在 “自动部署目录”中，如果指定建立应用程序的目录，则不必将文件复制到默认自动部署目录中。

   默认目录是服务器实例的根目录中名为 autodeploy 的目录。默认情况下，可使用变量自动更改多个服务器实例的目录。
4. 要在部署之前运行验证器，请选中 “已启用验证器”复选框。验证器将检查文件的结构和内容。大型应用程序的检验通常会很耗费时间。
5. 要预编译 JSP 页面，请选中 "JSP" 复选框。如果未选中此复选框，则首次访问 JSP 页面时会在运行时编译这些页面。由于编译通常很耗费时间，因此在生产环境中请选中此复选框。

设置其他属性

单击 “添加属性”按钮以指定其他设置。

设置域属性

包括以下域属性。

表 1-1 域属性值

属性	定义
com.sun.aas.installRoot	应用程序服务器的安装目录。
com.sun.aas.instanceRoot	服务器实例的顶层目录。
com.sun.aas.hostName	主机 （计算机）的名称。
com.sun.aas.javaRoot	.J2SE 安装目录。
com.sun.aas.imqLib	Sun Java System Message Queue 的库目录。
com.sun.aas.configName	服务器实例正在使用的配置的名称。
com.sun.aas.instanceName	服务器实例的名称。该属性对于 default-config 不可用，但可用于自定义的配置。
com.sun.aas.clusterName	群集的名称。仅在群集的服务器实例上设置了该属性。该属性对于 default-config 不可用，但可用于自定义的配置。
com.sun.aas.domainName	域的名称。该属性对于 default-config 不可用，但可用于自定义的配置。

实例特定的配置属性

实例特定的配置属性将覆盖此实例的值。

注 默认值在绑定至实例的配置中定义。

要将值恢复为默认值，请执行以下步骤：

1. 删除覆盖值。
2. 单击 “保存”。

如果未设置覆盖值，则将使用默认值。

添加或删除实例属性

要添加属性，请：

- 单击“添加属性”按钮以指定其他设置。

可以使用以下属性名称/值对来配置资源：

表 1-2 属性名称/值对

属性	定义
HTTP_LISTENER_PORT	此端口用于侦听 HTTP 请求。此属性指定 http-listener-1 的端口号。有效值为 1 到 65535。在 UNIX 中，创建在端口 1 到 1024 上进行侦听的套接字要求具有超级用户权限。
HTTP_SSL_LISTENER_PORT	此端口用于侦听 HTTPS 请求。此属性指定 http-listener-2 的端口号。有效值为 1 到 65535。在 UNIX 中，创建在端口 1 到 1024 上进行侦听的套接字要求具有超级用户权限。
IIOP_LISTENER_PORT	此属性指定 orb-listener-1 侦听 IIOP 连接的 ORB 侦听器端口。
IIOP_SSL_LISTENER_PORT	此端口用于安全 IIOP 连接。
JMX_SYSTEM_CONNECTOR_PORT	此属性指定 JMX 连接器进行侦听的端口号。有效值为 1 到 65535。在 UNIX 中，创建在端口 1 到 1024 进行侦听的套接字要求具有超级用户权限。
IIOP_SSL_MUTUALAUTH_PORT	此属性指定称为 SSL_MUTUALAUTH 的 IIOP 侦听器侦听 IIOP 连接的 ORB 侦听器端口。

要删除属性，请执行以下步骤：

1. 单击要删除的属性。
2. 单击“删除属性”按钮。

创建实例

要创建实例，请执行以下步骤：

1. 在树组件中，选择“独立实例”节点。
2. 在“独立服务器实例”页面中，单击“新建”。
3. 在“名称”字段中，标识新实例的唯一名称。

4. 选择一个节点代理。

   必须在节点代理的主机上使用 asadmin start-node-agent 命令启动节点代理，以便使正在创建的服务器实例能够与该节点代理相关联。

5. 选择所需的配置。

   - 引用现有配置。不添加新配置。
   - 创建现有配置的副本。添加服务器实例或群集时，将添加新配置。

6. 默认情况下，创建新实例时，其配置将从 default-config 配置复制而来。要从其他配置进行复制，请在创建新实例时指定要复制的配置。

7. 对于服务器实例，新配置的名称为 instance_name-config。

   default-config 配置为默认配置，它用作创建独立服务器实例的模板。不允许非群集服务器实例或群集引用 default-config 配置；只能复制该配置以创建新配置。编辑默认配置，以确保从默认配置复制而来的新配置具有正确的初始设置。

等效的 asadmin 命令为：create-instance。

启动实例

要启动实例，请执行以下步骤：

1. 在树组件中，展开 “独立实例” 节点。
2. 选择要启动的实例。
3. 在 “常规” 选项卡上，单击 “启动” 以启动实例。

必须先使用 asadmin start-node-agent 命令启动与实例相关联的节点代理，才能成功启动该实例。

启动实例后，便可以从 “常规” 选项卡上执行以下任务：

- 单击 “停止实例” 以停止实例。
- 单击 “JNDI 浏览” 以查看该实例的 JNDI 条目。
- 单击 “查看日志文件” 以查看 “日志查看器” 并指定日志记录选项。
- 单击 “轮转日志文件”。
- 单击 “恢复事务” 以恢复未完成的事务。

等效的 asadmin 命令为：start-instance。

恢复事务

由于服务器崩溃或资源管理器崩溃，事务可能未完成。完成这些被搁置的事务并将其从故障中恢复至关重要。Application Server 旨在在服务器启动时从故障中恢复并完成这些事务。

如果选定的服务器正在运行，则将由同一服务器执行恢复操作。如果选定的服务器没有运行，则将由选定的 Destination Server 执行恢复操作。

停止实例

要停止实例，请执行以下步骤：

1. 在树组件中，展开 “独立实例”节点。
2. 选择要停止的实例。
3. 在 “常规”选项卡上，单击 “停止”以停止实例。

等效的 asadmin 命令为：stop-instance。

关闭服务器实例

要关闭 Administration Server，请

1. 在树组件中，选择 “独立实例”节点。
2. 选择 “Administation Server 实例”
3. 单击 “停止”。

将显示确认对话框以确认您要关闭 Administration Server。

配置更改

- 更改 Application Server 配置
- Application Server 中的端口
- 查看端口号

- 更改管理服务器端口
- 更改 HTTP 端口
- 更改 IIOP 端口
- 使用管理服务配置 JMX 连接器
- 编辑 JMX 连接器配置
- 更改 J2SE 软件

更改 Application Server 配置

在进行以下任何配置更改时，请重新启动服务器以使更改生效：

- 更改 JVM 选项
- 更改端口号
- 管理 HTTP 服务、 IIOP 服务和 JMS 服务
- 管理线程池

有关说明请参见 “重新启动服务器或域”。

如果使用动态配置，大多数更改在服务器运行时即可生效。要进行以下配置更改，请**不要**重新启动服务器：

- 部署和取消部署应用程序
- 添加或删除 JDBC、 JMS 和连接器资源和池
- 更改日志记录级别
- 添加文件领域用户
- 更改监视级别
- 启用和禁用资源和应用程序

请注意， `asadmin reconfig` 命令已过时，并且不再需要此命令。配置更改将被动态应用到服务器。

Application Server 中的端口

表 1-3 介绍了 Application Server 的端口侦听器。

表 1-3 使用端口的 Application Server 侦听器

侦听器	默认端口号	说明
管理服务器	4848	通过管理控制台和 asadmin 实用程序访问域的管理服务器。对于管理控制台，请在浏览器的 URL 中指定端口号。远程执行 asadmin 命令时，请使用 --port 选项指定端口号。
HTTP	8081	Web 服务器侦听端口上的 HTTP 请求。要访问已部署的 Web 应用程序和服务，客户机应连接到此端口。
HTTPS	8181	为安全通信配置的 Web 应用程序在单独的端口上进行侦听。
IIOP		企业 Bean （EJB 组件）的远程客户机通过 IIOP 侦听器访问 Bean。
IIOP、 SSL		另一个端口由为安全通信配置的 IIOP 侦听器使用。
IIOP、 SSL 和双向验证		另一个端口由为双向 （客户机和服务器）验证配置的 IIOP 侦听器使用。

查看端口号

1. 在树组件中，选择 “独立实例”节点下的一个实例。
2. 选择 “属性”选项卡。
3. 在 “实例特性”页面中，标识了默认端口号。可以设置配置以覆盖这些值。

更改管理服务器端口

1. 在树组件中，展开 “配置”节点。
2. 展开 “server-config （管理配置）”节点
3. 展开 “HTTP 服务”节点。
4. 展开 “HTTP 侦听器”节点。
5. 选择 admin-listener 节点。
6. 在 “编辑 HTTP 侦听器”页面中，更改 “侦听器端口”字段的值。
7. 重新启动服务器。

更改 HTTP 端口

1. 在树组件中，展开 “HTTP 服务”节点。
2. 展开 “HTTP 侦听器”节点。
3. 选择要更改其端口号的 HTTP 侦听器。
4. 在 “编辑 HTTP 侦听器”页面中，更改 “侦听器端口”字段的值。
5. 单击 “保存”。
6. 重新启动服务器。

更改 IIOP 端口

1. 在树组件中，展开 “配置”节点。
2. 展开 “server-config （管理配置）”节点。
3. 展开 ORB 节点。
4. 展开 “IIOP 侦听器”节点。
5. 选择要更改其端口号的侦听器。
6. 在 “编辑 IIOP 侦听器”页面中，更改 “侦听器端口”字段的值。
7. 单击 “保存”。
8. 重新启动服务器。

使用管理服务配置 JMX 连接器

使用管理服务可以配置 JSR-160 兼容的远程 JMX 连接器，该连接器处理域管理服务器与用于远程服务器实例的节点代理 （管理主机上的服务器实例）之间的通信。

管理服务确定服务器实例是一个常规实例、一个域管理服务器 (DAS) 还是一个组合。 DAS 与 J2EE 服务器实例类似，只是用户应用程序和资源不会被部署到 DAS （尽管它能够处理用户应用程序请求）。 DAS 和 J2EE 服务器实例之间唯一比较显著的区别在于：前者不能是群集的一部分，而是服务器实例的同构单元。

要配置 JMX 连接器，请执行以下步骤：

1. 从树中选择 “配置”。
2. 2. 选择要配置的实例：
    a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 "server-config" 节点。
    b. b. 要为将来要使用 default-config 的副本的实例配置默认设置，请选择 default-config 节点。
3. 从树中选择 “管理服务”。
4. 从 “类型”下拉式菜单中，选择希望管理服务配置的类型：DAS、 DAS 和服务器或者服务器。选择 “DAS 和服务器”与选择 "DAS" 相同。“服务器”选项将选择非 DAS 服务器实例。
5. 在 “JMX 连接器名称”字段中，输入内部使用的 JMX 连接器的名称。连接器的名称为 system。

编辑 JMX 连接器配置

使用 “编辑 JMX 连接器”屏幕，您可以编辑 JSR 160 兼容的 JMX 连接器的配置。

1. 从树中选择 “配置”。
2. 选择要配置的实例：
    a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 "server-config" 节点。
    b. 要为将来的实例 （使用 default-config 的副本）配置默认设置，请选择 "default-config" 节点。
3. 展开 “管理服务”节点并单击 “系统”，它是内部使用的 JMX 连接器。
4. 输入 JMX 连接器服务器的端口。 JMX 服务 URL 是协议、端口和地址的函数，如 JSR 160 1.0 规范所定义
5. 输入此 JMX 连接器应该支持的协议。 Application Server 版本 8.1 仅支持 rmi_jrmp 协议。
6. 在 “领域名称”字段中，输入表示特殊管理领域的名称。此区域将处理所有验证。
7. 选中 “已启用”复选框以表明 JMX 连接器中应当使用传输层安全性。如果启用传输层安全性，请按照 “配置管理服务的 JMX 连接器的安全性”中的说明填写 SSL 部分。

更改 J2SE 软件

Application Server 依赖于 Java 2 Standard Edition (J2SE) 软件。安装 Application Server 时，指定 J2SE 软件的目录。有关更改 J2SE 软件的说明，请参见 “配置 JVM 常规设置”。

使用联机帮助

管理控制台的联机帮助是上下文相关的：单击右上角的 “帮助”链接时，帮助浏览器窗口会显示与当前管理控制台页面相关的主题。如果当前页面没有帮助信息，则显示 “使用联机帮助”主题。

联机帮助包含了上下文不相关的概念主题。要查看这些主题中的某一个主题，请从帮助浏览器窗口的目录中选择该主题。

要返回到上一帮助屏幕，请执行以下步骤：

1. 在帮助浏览器窗口中，单击鼠标右键以显示选择菜单。
2. 选择 “后退”。

如果没找到要查找的信息，请参见 Application Server Administration Guide，可从以下网址获得：

```
http://docs.sun.com/
```

更多信息

- Sun Microsystems 全球培训－Sun 及其授权中心每年通过基于 Web 的课程和遍布 60 多个国家 / 地区的 250 多个培训站点培训超过 250,000 名学员。有关更多信息，请访问：

  ```
  http://training.sun.com/
  ```

- 《 The J2EE 1.4 Tutorial 》－该联机教程专为开发者编写，其中介绍了配置 JMS、设置 JavaMail 资源和管理安全性的管理说明。要访问此联机教程，请转至下面的 URL：

  ```
  http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html
  ```

- Application Server Developer’s Guide －该指南提供了专用于 Application Server 的开发信息。 Developer’s Guide 可从以下网址获得：

  ```
  http://docs.sun.com/
  ```

- asadmin 手册页－以 HTML 格式提供，这些手册页包含所有应用程序服务器实用程序 （包括 asadmin 实用程序命令）的语法和示例。这些 HTML 页面发布在以下 URL：

  http://docs.sun.com/

- Application Server 发行说明－可从以下网址获得：

  http://docs.sun.com/

- Getting Started With J2EE Connectors －该文档提供了有关配置连接器 （资源适配器）、连接池和连接器资源的说明：

  http://java.sun.com/j2ee/connector/

- docs.sun.com：Sun 产品文档－从该站点您可以搜索和访问我们产品的所有文档：

  http://docs.sun.com/

- J2EE 1.4 Documentation 页面－位于我们的公用 Web 站点上，该页面具有指向 J2EE 1.4 平台的技术文档的链接：

  http://java.sun.com/j2ee/1.4/docs/

- 快速入门指南－该文档介绍了如何部署和运行一个简单 Web 应用程序。该指南位于 *install_dir*/docs/QuickStart.html 文件中。

第 2 章

配置群集

本章介绍了如何使用管理控制台来配置群集。它包含以下各节：

关于群集

什么是群集？

群集是作为一个逻辑实体一起工作的零个或多个服务器实例的集合。群集为一个或多个 J2EE 应用程序提供了运行时环境。

在 Sun Java System Application Server Enterprise Edition 8.1 环境中使用群集具有以下优点：

- 高可用性 （通过允许为群集中的服务器实例提供故障转移保护来实现）。如果一个服务器实例出现故障，其他服务器实例将接管该服务器实例正在服务的请求。
- 可伸缩性 （通过允许向群集中添加服务器实例从而增加系统的容量来实现）。Application Server 的负载平衡器会将请求分布到群集中的可用服务器实例。当管理员向群集中添加更多服务器实例时，无需中断服务。

群集具有以下特性：

- 群集中的所有实例都引用相同的配置。
- 群集中的所有实例都具有相同的一组已部署的应用程序（例如，J2EE 应用程序 EAR 文件、 Web 模块 WAR 文件或 EJB JAR 文件）。
- 群集中的所有实例都具有相同的资源组，因此它们都具有相同的 JNDI 名称空间。

域中的每一个群集都具有唯一的名称；此外，该名称在所有节点代理名称、服务器实例名称、群集名称和配置名称中也必须是唯一的。此名称不能为 `domain`。管理员在群集上执行的操作与在非群集服务器实例上执行的操作相同操作 （例如，部署应用程序和创建资源）。

有关群集、节点代理和服务器实例的使用的概述，请参见 Deployment Planning Guide。有关负载平衡器的详细信息，请参见 “配置负载平衡和故障转移”。

群集类型

有两种群集类型：**独立群集**和**共享群集**。

- 独立群集具有自己的配置，它不与其他服务器实例或群集共享其配置。默认情况下，此配置的名称为 *cluster_name*`-config`，其中 *cluster_name* 表示群集的名称。
- 共享群集与一个或多个其他群集或非群集实例共享其配置。

群集、实例、负载平衡器和会话

群集、服务器实例、负载平衡器和会话在 Application Server 中的相互关系如下：

- 服务器实例可以不属于群集。但是，不属于群集的实例无法通过将会话状态从一个实例转移到其他实例来利用高可用性。
- 群集中的服务器实例可以位于不同的计算机上，也可以位于同一计算机上。也就是说，可以将不同计算机上的服务器实例组织为一个群集。
- 特定负载平衡器可以向多个群集中的服务器实例转发请求。您可以使用负载平衡器的此功能来执行联机升级，而不使服务受到损失。有关更多信息，请参见 第 58 页上的 “将多个群集用于联机升级而不使服务受到损失”。

- 一个群集可以从多个负载平衡器接收请求。如果群集由多个负载平衡器提供服务，则必须以完全相同的方式在每个负载平衡器上配置群集。
- 每个会话都依赖于特定的群集。这意味着尽管可以将应用程序部署在多个群集中，但会话将只能被故障转移到同一群集中的服务器实例。
- Application Server 支持 HTTP 会话和有状态会话 Bean (SFSB) 会话的故障转移。也支持某些存储在 HTTP 会话中的 J2EE 对象引用的故障转移。

  有关 HTTP 会话故障转移的更多信息，请参见 第 143 页上的 “配置可用性和会话持久性”

因此，对于群集中的服务器实例，群集充当的是会话故障转移的安全边界。在 Application Server 中，您可以使用负载平衡器和升级组件，而不使服务受到任何损失。有关更多信息，请参见第 58 页上的 “将多个群集用于联机升级而不使服务受到损失”。

用于群集的管理控制台任务

- 创建群集
- 配置群集
- 移植 EJB 计时器
- 为群集创建服务器实例
- 配置群集服务器实例
- 为群集配置应用程序
- 为群集配置资源
- 删除群集
- 将多个群集用于联机升级而不使服务受到损失

创建群集

要创建群集，请执行以下步骤：

1. 在树组件中，选择 “群集”节点。
2. 在 “群集”页面中，单击 “新建”。将显示 “创建群集”页面。
3. 在 “名称”字段中，键入群集的名称。此名称
   - 只能由大写和小写字母、数字、下划线、连字符和句点 (.) 组成
   - 在所有节点代理名称、服务器实例名称、群集名称和配置名称中都必须是唯一的
   - 不能为 `domain`
4. 在 “配置”字段中，从下拉式列表中选择配置。

   要创建独立群集，请选择 `default-config`，并选中标有 “复制选定的配置”的单选按钮。默认配置的副本的名称将为 *cluster_name*`-config`。

   要引用其他配置，请从下拉式列表中选择配置，并选择标有 “引用选定的配置”的单选按钮。如果另一个群集正在使用该配置，则此操作将创建共享群集。
5. 您可以现在添加服务器实例，也可以在创建群集后再添加服务器实例。为群集添加服务器实例之前，请先创建一个或多个节点代理或节点代理占位符。有关详细信息，请参见第 98 页上的 “创建节点代理占位符”。

   要创建服务器实例，请执行以下步骤：

   a. 在 “要创建的服务器实例”区域，单击 “添加”。

   b. 在 “实例名称”字段中为实例输入名称。

   c. 从 “节点代理”下拉式列表中选择节点代理。
6. 单击 “确定”。
7. 在显示的 “已成功创建群集”页面中单击 “确定”。

有关如何管理群集、服务器实例和节点代理的详细信息，请参见第 91 页上的 “部署节点代理”。

等效的 `asadmin` 命令为：`create-cluster`

配置群集

要配置群集，请执行以下步骤：

1. 在树组件中，展开 “群集”节点。
2. 选择群集的节点。在 “常规信息”页面中，您可以执行以下任务：
   - 单击 “启动实例”以启动群集服务器实例。
   - 单击 “停止实例”以停止群集服务器实例。
   - 单击 “移植 EJB 计时器”以将 EJB 计时器从已停止的服务器实例移植到群集中的其他服务器实例。

等效的 asadmin 命令为：start-cluster、 stop-cluster、 migrate-timers

移植 EJB 计时器

如果服务器实例非正常或意外地停止运行，则可能需要将该服务器实例上安装的 EJB 计时器移至群集中正在运行的服务器实例。要完成此操作，请执行以下步骤：

1. 从 “源”下拉式列表中，选择要移植的计时器所在的已停止的服务器实例。
2. 从 “目标”下拉式列表中，选择要将计时器移植到的正在运行的服务器实例（此操作可选）。如果将该字段保留为空，将随机选择一个正在运行的服务器实例。
3. 单击 “确定”。
4. 停止并重新启动目标服务器实例。

如果源服务器实例正在运行或目标服务器实例未运行，将显示错误消息。

等效的 asadmin 命令为：migrate-timers

为群集创建服务器实例

为群集创建服务器实例之前，必须先创建节点代理或节点代理占位符。有关详细信息，请参见第 98 页上的 “创建节点代理占位符”。

要为群集创建服务器实例，请执行以下步骤：

1. 在树组件中，展开 “群集”节点。
2. 选择群集的节点。
3. 单击 “实例”选项卡以显示 “群集服务器实例”页面。
4. 单击 “新建”以显示 “创建群集服务器实例”页面。
5. 在 “名称”字段中，键入服务器实例的名称。
6. 从 “节点代理”下拉式列表中选择节点代理。
7. 单击 “确定”。

等效的 `asadmin` 命令为：`create-instance`

配置群集服务器实例

要在创建群集服务器实例后对其进行更改，请执行以下步骤：

1. 在树组件中，展开 “群集”节点。
2. 展开包含服务器实例的群集的节点，然后选择要进行编辑的服务器实例节点。
3. 在 “常规信息”页面中，您可以执行以下任务：
   - 单击 “启动实例”以启动实例。
   - 单击 “停止实例”以停止正在运行的实例。
   - 单击 “JNDI 浏览”以浏览正在运行的实例的 JNDI 树。
   - 单击 “查看日志文件”，以打开服务器日志查看器。
   - 单击 “轮转日志文件”，以轮转实例的日志文件。该操作将安排日志文件以进行轮转。实际的轮转将在下一次向日志文件写入条目时发生。
   - 单击 “恢复事务”以恢复未完成的事务。
   - 单击 “属性”选项卡，以修改实例的端口号。
   - 单击 “监视”选项卡，以更改监视属性。

也可以在服务器实例上执行以下操作：

1. 在树组件中，展开 “群集”节点。
2. 展开包含服务器实例的群集的节点。
3. 单击 “实例”选项卡以转至 “群集服务器实例”页面。在此页面中，您可以执行以下任务：
   - 选中实例的复选框，并单击 “删除”、“启动”或 “停止”。
   - 单击实例的名称，以显示 “常规信息”页面。

要从群集中删除群集服务器实例，请选中实例名称旁边的复选框并单击 “删除”。

为群集配置应用程序

要为群集配置应用程序，请执行以下步骤：

1. 在树组件中，展开 “群集”节点。
2. 选择群集的节点。
3. 单击 “应用程序”选项卡，以显示 “应用程序”页面。在此页面中，您可以执行以下任务：
   - 选中应用程序旁边的复选框，然后选择 “启用”或 “禁用”以启用或禁用用于群集的应用程序。
   - 从 “部署”下拉式列表中，选择要部署的应用程序的类型。在显示的 “部署”页面中，指定应用程序。
   - 从 “过滤器”下拉式列表中，选择要在列表中显示的应用程序的类型。

要编辑应用程序，请单击应用程序名称。

为群集配置资源

要为群集配置资源，请执行以下步骤：

1. 在树组件中，展开 “群集”节点。
2. 选择群集的节点。

3. 单击 “资源”选项卡，以显示 “资源”页面。在此页面中，您可以执行以下任务：
   - 选中资源旁边的复选框，然后选择 “启用”或 “禁用”以全局启用或禁用资源。该操作不会删除资源。
   - 从 “新建”下拉式列表中，选择要创建的资源的类型。创建资源时，请确保将群集指定为目标。
   - 从 “过滤器”下拉式列表中，选择要在列表中显示的资源的类型。

要编辑资源，请单击资源名称。

删除群集

要删除群集，请执行以下步骤：

1. 在树组件中，选择 “群集”节点。
2. 在 “群集”页面中，选中要删除的群集的名称旁边的复选框。
3. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-cluster`

将多个群集用于联机升级而不使服务受到损失

在 Application Server 中，您可以使用负载平衡器和多个群集来升级组件，而不使服务受到任何损失。例如，组件可以是 JVM、 Application Server 或 Web 应用程序。

要执行此任务，请执行以下操作：

1. 使用群集的 “常规信息”页面上的 “停止群集”按钮来停止其中一个群集。
2. 升级该群集中的组件。
3. 使用群集的 “常规信息”页面上的 “启动群集”按钮来启动群集。
4. 对其他群集逐个重复此过程。

由于一个群集中的会话决不会故障转移到另一个群集中的会话，因此不会出现由会话故障转移所引起的版本不匹配，这种故障转移是指从运行一个版本的组件的服务器实例故障转移到运行其他版本的组件的另一个服务器实例 （位于其他群集中）。这样，群集就用作了它所包含的服务器实例的会话故障转移的安全边界。

注 在以下情况下不能使用此方法：

- 当您更改高可用性数据库 (HADB) 的模式时。有关更多信息，请参见 《Sun Java System Application Server High Availability Administration Guide》中的 "Administering High Availability Database" 一章。
- 当您执行涉及对应用程序数据库模式的更改的应用程序升级时。

警告 请同时升级群集中的所有服务器实例。否则，可能会出现由从运行不同版本的组件的一个实例到另一个实例的会话故障转移引起的版本不匹配。

第 3 章

配置负载平衡和故障转移

本章介绍了如何设置 Sun Java System Application Server 中的 HTTP 请求的负载平衡。并说明了如何配置由负载平衡器所控制的服务器实例之间的故障转移。此外，本章还讨论了 RMI-IIOP 负载平衡和故障转移。

它包含以下各节：

关于 HTTP 负载平衡和故障转移

HTTP 负载平衡和故障转移

负载平衡的目的是在多个 Sun Java System Application Server 实例 （独立或群集）之间平均分配工作量，从而提高系统的整体吞吐量。

使用负载平衡器还可以启用从一个服务器实例故障转移到另一个服务器实例的请求。对于要保留的 HTTP 会话信息，请配置 HTTP 会话持久性。有关更多信息，请参见 “配置可用性和会话持久性”。

使用 asadmin 工具而不是管理控制台来配置 HTTP 负载平衡。

HTTP 负载平衡的要求

在使用负载平衡器插件进行 HTTP 请求之前，必须满足以下要求：

- 已安装 Sun Java System Application Server Enterprise Edition。
- 安装并配置了 Web 服务器。

  有关更多信息，请参见第 65 页上的 “为 HTTP 负载平衡配置 Web 服务器”。
- 已安装负载平衡器插件。
- 已配置 Application Server。
  - 已创建参与负载平衡的 Application Server 实例或群集。
  - 应用程序已部署到参与负载平衡的所有 Application Server 实例或群集中。
  - 参与负载平衡的服务器实例和群集必须具有同构环境。通常，这意味着服务器实例将引用相同的服务器配置，并且部署到服务器实例中的应用程序也相同。

了解指定的请求和未指定的请求

在某个请求首次从 HTTP 客户机传入负载平衡器时，该请求为新会话请求。新会话请求称为**未指定的**请求。负载平衡器会根据 Round Robin 算法将此请求路由到群集中的应用程序服务器实例。有关更多信息，请参见第 63 页上的 “HTTP 负载平衡算法”。

在某个应用程序服务器实例中创建会话后，负载平衡器会将此会话的所有后续请求都路由到该特定实例而且只路由到该实例。现有会话的请求称为**指定的**或**粘性**请求。

HTTP 负载平衡算法

Sun Java System Application Server 负载平衡器使用粘性 Round Robin 算法对收到的 HTTP 和 HTTPS 请求进行负载平衡。给定会话的所有请求都将被发送到同一个应用程序服务器实例。使用粘性负载平衡器，会话数据将被高速缓存在单个应用程序服务器上，而不会被分布到群集中的所有实例。

因此，粘性 Round Robin 方案能够带来明显的性能优势，这些优势超过了使用纯 Round Robin 带来的更加平均分布负载的优势。

关于粘性 Round Robin 负载平衡算法

当新的 HTTP 请求发送到负载平衡器插件时，系统将基于简单的 Round Robin 方案将该请求转发到某个应用程序服务器实例；随后，将通过使用 Cookie 或显式 URL 重写将该请求 “粘”到此特定应用程序服务器实例上。

从粘性信息中，负载平衡器插件将首先确定请求先前被转发到的实例。如果发现该实例工作正常，负载平衡器插件会将请求转发至该特定应用程序服务器实例。因此，给定会话的所有请求都将被发送到同一个应用程序服务器实例。

负载平衡器插件使用以下方法来确定会话粘性：

- 基于 Cookie 的方法
- 显式 URL 重写方法

基于 Cookie 的方法

在基于 Cookie 的方法中，负载平衡器插件使用一个单独的 Cookie 来记录路由信息。

注	要使用基于 Cookie 的方法， HTTP 客户机必须支持 Cookie。

显式 URL 重写方法

在显式 URL 重写方法中，粘性信息将被附加至 URL。即使 HTTP 客户机不支持 Cookie，也可以使用此方法。

负载平衡和故障转移样例应用程序

以下目录包含了用于演示负载平衡和故障转移的样例应用程序：

install_dir/samples/ee-samples/highavailability
install_dir/samples/ee-samples/failover

ee-samples 目录还包含有关设置运行样例的环境的信息。

HTTP 负载平衡设置概述

使用 asadmin 工具可以在您的环境中配置负载平衡。请执行以下步骤：

1. 完成第 62 页上的 “HTTP 负载平衡的要求”，包括安装和配置 Web 服务器和 Application Server 实例和 （或）群集。
2. 使用 asadmin 命令 create-http-lb-config 创建负载平衡器配置。
3. 使用 asadmin create-http-lb-ref 为要管理的负载平衡器添加对群集和独立服务器实例的引用。

   如果您创建了具有目标的负载平衡器配置，并且该目标是负载平衡器引用的唯一群集或独立服务器实例，则请跳过此步骤。
4. 使用 asadmin enable-http-lb-server 启用负载平衡器引用的群集或独立服务器实例。
5. 使用 asadmin enable-http-lb-application 启用要用于负载平衡的应用程序。

   这些应用程序必须已部署到负载平衡器所引用的群集或独立实例上，并且已启用，可以在群集或独立实例上使用。启用应用程序以用于负载平衡与启用以使用这些应用程序是两个独立的步骤。
6. 使用 asadmin create-health-checker 创建运行状况检查器。

   运行状况检查器监视工作异常的服务器实例，以便在这些服务器实例重新正常工作时，负载平衡器可以向它们发送新请求。
7. 使用 asadmin export-http-lb-config 生成负载平衡器配置文件。

   此命令用于生成要同 Sun Java System Application Server 附带的负载平衡器插件一起使用的配置文件。
8. 将负载平衡器配置文件复制到 Web 服务器的 config 目录，此目录中存储了负载平衡器插件配置文件。

为 HTTP 负载平衡配置 Web 服务器

关于 Web 服务器配置

负载平衡器插件安装程序将对 Web 服务器的配置文件进行一些修改。所作的更改取决于该 Web 服务器。

注 在运行支持的 Web 服务器的计算机上，负载平衡器插件可随 Sun Java System Application Server Enterprise Edition 一起安装，也可单独安装。

有关安装过程的完整信息，请参见 Sun Java System Application Server 安装指南。

对 Sun Java System Web Server 的修改

安装程序将对 Sun Java System Web Server 的配置文件进行以下更改：

1. 将以下负载平衡器插件特定的条目添加到 Web 服务器实例的 `magnus.conf` 文件中：

```
##EE lb-plugin
Init fn="load-modules"
shlib="web_server_install_dir/plugins/lbplugin/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrough"
Thread="no"

Init fn="init-passthrough"

##end addition for EE lb-plugin
```

2. 将以下特定于负载平衡器插件的条目添加到 Web 服务器实例的 obj.conf 文件中：

```
<Object name=default>

NameTrans fn="name-trans-passthrough" name="lbplugin"
config-file="web_server_install_dir/web_server_instance/config/loadbalancer.xml"

<Object name="lbplugin">
ObjectType fn="force-type" type="magnus-internal/lbplugin"
PathCheck fn="deny-existence" path="*/WEB-INF/*"
Service type="magnus-internal/lbplugin" fn="service-passthrough"
Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</object>
```

lbplugin 是唯一标识 Object 的名称；*web_server_install_dir*/*web_server_instance*/config/loadbalancer.xml 是负载平衡器要在其上运行的虚拟服务器的 XML 配置文件所在的位置。

安装后，请按照第 64 页上的 “HTTP 负载平衡设置概述”中的说明配置负载平衡器。

对 Apache Web Server 的修改

在 Apache 上安装负载平衡器插件之前，请参见附录 A “编译和配置 Apache Web Server”中有关编译和配置 Apache 的信息。

安装程序所作的修改

负载平衡器插件安装程序将所需文件解压缩到 Web 服务器根目录下的 libexec (Apache 1.3) 或 modules (Apache 2.0) 文件夹下。它将以下特定于负载平衡器插件的条目添加到 Web 服务器实例的 httpd.conf 文件中：

```
<VirtualHost machine_name:443>

##Addition for EE lb-plugin

LoadFile /usr/lib/libCstd.so.1

LoadModule apachelbplugin_module libexec/mod_loadbalancer.so
#AddModule mod_apachelbplugin.cpp
<IfModule mod_apachelbplugin.cpp>
        config-file webserver_instance/conf/loadbalancer.xml
locale en
</IfModule>
```

```
<VirtualHost machine_ip_address>
DocumentRoot "webserver_instance/htdocs"
ServerName server_name
</VirtualHost>

##END EE LB Plugin ParametersVersion 7
```

注

- 在 Apache 1.3 上，当多个 Apache 子进程运行时，每个进程都有自己的负载平衡 Round Robin 序列。

  例如，如果有两个 Apache 子进程正在运行，并且负载平衡插件对两个应用程序服务器实例进行负载平衡，则第一个请求将发送给实例 1，第二个请求也将发送给实例 1，而第三个请求将发送给实例 2，第四个请求也将发送给实例 2。系统将重复执行这种模式 （实例 1、实例 1、实例 2、实例 2 等）。

  此行为可能与您预期的行为 （即，实例 1、实例 2、实例 1、实例 2 等）不同。在 Sun Java System Application Server 中，用于 Apache 的负载平衡插件将为每个 Apache 进程启动一个负载平衡器实例，从而创建独立的负载平衡序列。

- 如果使用 `--with-mpm=worker` 选项进行编译，则 Apache 2.0 将具有多线程行为。

安装后的修改

Apache Web 服务器必须在安全模式下运行，以确保它能够使用负载平衡器插件正常进行工作。请在 *apache_install_dir* 下创建名为 `sec_db_files` 的目录并将 *application_server_domain_dir*`/config/`*security_db_files* 复制到 *apache_install_dir*`/`*sec_db_files*。

对 Microsoft Windows 的其他修改

如果是在 Microsoft Windows 上运行 Apache，则在安装插件后，需要更改某些环境变量：

通过单击 “开始” -> “设置” -> “控制面板” -> “系统” -> “高级” -> “环境变量” -> “系统变量” 将新路径添加到 Path 环境变量中。编辑 Path 变量使其包含以下路径：

application_server_install_dir`/bin`

此外，在启动 Apache Web 服务器之前，请将环境变量 NSPR_NATIVE_THREADS_ONLY 设置为 1。

在 “环境变量”窗口的 “系统变量”下，单击 “新建”。输入以下名称和值对：

变量名：NSPR_NATIVE_THREADS_ONLY

变量值：1

重新启动计算机。

对 Microsoft IIS 的修改

要配置 Microsoft Internet 信息服务 (IIS) 以使用负载平衡器插件，请修改 Windows Internet 服务管理器中的某些属性。 Internet 服务管理器位于 “控制面板”文件夹的 “管理工具”文件夹中。

安装 Sun Java System Application Server 后，请执行以下修改。

1. 打开 Internet 服务管理器。
2. 选择要为其启用插件的 Web 站点。此 Web 站点通常命名为 “默认的 Web 站点”。
3. 在此 Web 站点上单击鼠标右键，并选择 “属性”以打开 “属性”记事本。
4. 要添加新的 ISAPI 过滤器，请打开 “ISAPI 过滤器”选项卡，单击 “添加”，然后执行以下步骤：
   a. 在 “过滤器名”字段中，输入 Application Server
   b. 在 “可执行文件”字段中，键入 `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.dll`
   c. 单击 “确定”关闭 “属性”记事本。
5. 创建并配置新的虚拟目录：
   a. 在默认的 Web 站点上单击鼠标右键，选择 “新建”，然后选择 “虚拟目录”。

      将打开 “虚拟目录创建向导”。
   b. 在 “别名”字段中，键入 `sun-passthrough`。
   c. 在 “目录”字段中，键入 `C:\Inetpub\wwwroot\sun-passthrough`
   d. 选中 “执行权限”复选框。使与权限相关的所有其他复选框保持未选中状态。
   e. 单击 “完成”。

6. 将 sun-passthrough.dll 文件的路径和 *application_server_install_dir*/bin 添加到系统的 PATH 环境变量中。重新启动计算机。

7. 停止然后启动 Web 服务器以使新设置生效。

   要停止 Web 服务器，请在此 Web 站点上单击鼠标右键，然后选择 “停止”。要启动 Web 服务器，请在此 Web 站点上单击鼠标右键，然后选择 “启动”。

   接下来，在 Web 浏览器中键入以下地址，以访问此 Web 应用程序的上下文根：

   http://*webserver_name*/*web_application*

   其中， *webserver_name* 是 Web 服务器的主机名或 IP 地址， */web_application* 是 C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties 文件中列出的上下文根路径。检验 Web 服务器、负载平衡器插件和 Application Server 是否运行正常。

安装程序将自动配置 sun-passthrough.properties 中的下列属性。您可以更改默认值。

表 3-1 为 Microsoft IIS 自动配置的 sun-passthrough.properties

属性	定义	默认值
lb-config-file	负载平衡器配置文件的路径	*IIS_www_root*\sun-passthrough\loadbalancer.xml
log-file	负载平衡器日志文件的路径	*IIS_www_root*\sun-passthrough\lb.log
log-level	Web 服务器的日志级别	INFO

配置多个 Web 服务器实例

Sun Java System Application Server 安装程序不允许在单个计算机上安装多个负载平衡器插件。要在单个群集或多个群集中的单个计算机上安装多个带有负载平衡器插件的 Web 服务器，需要手动执行一些步骤来配置负载平衡器插件。

1. 配置新的 Web 服务器实例以使用负载平衡器插件，如第 65 页上的 “对 Sun Java System Web Server 的修改”、第 66 页上的 “对 Apache Web Server 的修改”或第 68 页上的 “对 Microsoft IIS 的修改”所述。

2. 将 sun-loadbalancer_1_1.dtd 文件从现有 Web 服务器实例的 config 目录复制到新实例的 config 目录中。

3. 要使用相同的负载平衡器配置，请将 loadbalancer.xml 文件从现有的 Web 服务器实例的 config 目录复制到新实例的 config 目录中。

4. 要使用其他负载平衡器配置，请执行以下步骤：

   a. 使用 `asadmin create-http-lb-config` 创建新的负载平衡器配置。

   b. 使用 `asadmin export http-lb-config` 将新配置导出到 `loadbalancer.xml` 文件中。

   c. 将 `loadbalancer.xml` 文件复制到新 Web 服务器的 `config` 目录中。

   有关创建负载平衡器配置并将其导出到 `loadbalancer.xml` 文件的信息，请参见“HTTP 负载平衡器配置任务”。

HTTP 负载平衡器配置任务

- 创建 HTTP 负载平衡器配置
- 创建 HTTP 负载平衡器引用
- 启用用于负载平衡的服务器实例
- 启用用于负载平衡的应用程序
- 创建 HTTP 运行状况检查器
- 导出负载平衡器配置文件
- 更改 HTTP 负载平衡器配置
- 启用动态重新配置
- 禁用 （停止）服务器实例或群集
- 禁用 （停止）应用程序
- 配置 HTTP 和 HTTPS 会话故障转移
- 配置幂等 URL
- 配置 HTML 错误页面

创建 HTTP 负载平衡器配置

负载平衡器配置是定义负载平衡器的 `domain.xml` 文件中的命名配置。

负载平衡配置非常灵活：

- 尽管每个负载平衡器只有一个负载平衡器配置，但每个负载平衡器配置可以关联多个负载平衡器。
- 尽管一个域可以关联多个负载平衡器，但一个负载平衡器只为一个域提供服务。

使用 `asadmin` 命令 `create-http-lb-config` 创建配置。指定下列参数：

- response timeout

  服务器实例必须返回响应的时间 （以秒为单位）。如果在该时间段内未收到任何响应，则服务器将被视为处于异常状态。默认值为 60。

- HTTPS routing

  指定对负载平衡器的 HTTPS 请求是否会导致对服务器实例的 HTTPS 或 HTTP 请求。

  有关更多信息，请参见第 77 页上的 “配置 HTTP 和 HTTPS 会话故障转移”。

- reload interval

  检查负载平衡器配置文件 `loadbalancer.xml` 的更改的时间间隔。当检查检测到更改时，系统将重新装入配置文件。 0 值禁用重新装入。

  有关更多信息，请参见第 75 页上的 “启用动态重新配置”。

- monitor

  指定是否为负载平衡器启用监视功能。

  有关更多信息，请参见第 79 页上的 “监视 HTTP 负载平衡器插件”。

- routecookie

  指定负载平衡器插件用于记录路由信息的 Cookie 的名称。 HTTP 客户机必须支持 Cookie。如果您的浏览器设置为在存储 Cookie 之前进行询问，则 Cookie 的名称为 JROUTE。

- target

  为负载平衡器配置指定目标。指定目标的效果与向该目标添加引用的效果相同。目标可以是群集，也可以是独立实例。

有关更多信息，请参见 `create-http-lb-config`、 `delete-http-lb-config` 和 `list-http-lb-configs` 的文档。

创建 HTTP 负载平衡器引用

当您在负载平衡器中创建对独立服务器或群集的引用时，该服务器或群集将被添加到负载平衡器控制的目标服务器和群集的列表中。仍然需要先启用所引用的服务器或群集 （使用 enable-http-lb-server），然后才能对该服务器或群集的请求进行负载平衡。如果创建了带有目标的负载平衡器配置，则系统已将该目标添加为引用。

使用 create-http-lb-ref 创建引用。您必须提供负载平衡器配置名称和目标服务器实例或群集。

要删除引用，请使用 delete-http-lb-ref。要删除某个引用，必须先使用 disable-http-lb-server 禁用引用的服务器或群集。

有关更多信息，请参见 create-http-lb-ref 和 delete-http-lb-ref 的文档。

启用用于负载平衡的服务器实例

创建对服务器实例或群集的引用后，请使用 enable-http-lb-server 启用服务器实例或群集。如果在创建负载平衡器配置时使用了某个服务器实例或群集作为目标，则必须启用该服务器实例或群集。

有关更多信息，请参见 enable-http-lb-server 的文档。

启用用于负载平衡的应用程序

负载平衡器管理的所有服务器都必须具有同构配置，包括部署到这些服务器的相同应用程序集。部署和启用某个应用程序以便进行访问 （在部署期间或部署之后发生）后，您必须启用该应用程序以进行负载平衡。如果没有为负载平衡启用应用程序，则即使已对该应用程序所部署到的服务器的请求执行了负载平衡和故障转移，也不会对该应用程序的请求执行负载平衡和故障转移。

启用应用程序时，请指定应用程序名称和目标。如果负载平衡器管理了多个目标（例如，两个群集），请在所有目标上启用该应用程序。

有关更多信息，请参见 enable-http-lb-application 的联机帮助。

如果部署了新的应用程序，则还必须启用该应用程序以进行负载平衡并再次导出负载平衡器配置。

创建 HTTP 运行状况检查器

负载平衡器的运行状况检查器将定期检查所有标记为异常的已配置 Application Server 实例。运行状况检查器不是必需的，但如果没有运行状况检查器，或者禁用了运行状况检查器，则不会执行异常实例的定期运行状况检查。

负载平衡器的运行状况检查机制使用 HTTP 与应用程序服务器实例进行通信。运行状况检查器将 HTTP 请求发送给指定的 URL 并等待响应。HTTP 响应标题中的状态码在 100 到 500 之间时表示实例处于正常状态。

创建运行状况检查器

要创建运行状况检查器，请使用 `asadmin` 的 `create-http-health-checker` 命令。指定下列参数：

- url

  指定负载平衡器检查的侦听器的 URL 以确定其运行状况。默认值为 `"/"`。

- `interval`

  指定进行实例的运行状况检查的时间间隔 （以秒为单位）。默认值为 30 秒。指定 0 将禁用运行状况检查器。

- timeout

  指定超时间隔 （以秒为单位），必须在该时间间隔内获得响应才能认为侦听器运行正常。默认值为 `10` 秒。

如果应用程序服务器实例被标记为异常，运行状况检查器将轮询异常实例以确定实例的状态是否已变为正常。运行状况检查器使用指定的 URL 来检查所有异常的应用程序服务器实例，以确定这些异常的应用程序服务器实例是否已返回到正常状态。

如果运行状况检查器发现某个异常实例已变为正常，该实例将被添加到正常实例列表中。

有关更多信息，请参见 `create-http-health-checker` 和 `delete-http-health-checker` 的文档。

正常实例的附加运行状况检查属性

`create-http-health-checker` 创建的运行状况检查器仅检查异常实例。要定期检查正常实例，请在导出的 `loadbalancer.xml` 文件中设置某些附加属性。

注 只能在导出 `loadbalancer.xml` **之后**对该文件进行手动编辑来设置这些属性。没有等效的 asadmin 命令可以使用。

要检查正常的实例，请设置以下属性：

表 3-2 运行状况检查器属性

属性	定义
active-healthcheck-enabled	True/False 标志，用于表示是否要对正常服务器实例执行 Ping 操作以确定这些实例是否正常。要对服务器实例执行 Ping 操作，请将标志设置为 True。
number-healthcheck-retries	指定在将未响应的服务器实例标记为异常之前，负载平衡器的运行状况检查器执行 Ping 操作的次数。有效范围在 1 到 1000 之间。默认设置值为 3。

通过编辑 `loadbalancer.xml` 文件来设置属性。例如：

```
<property name="active-healthcheck-enabled" value="true"/>

<property name="number-healthcheck-retries" value="3"/>
```

如果添加了这些属性，随后编辑并再次导出了 `loadbalancer.xml` 文件，则必须再次将这些属性添加到该文件中，因为新导出的配置不会包含这些属性。

导出负载平衡器配置文件

Sun Java System Application Server 附带的负载平衡插件使用名为 `loadbalancer.xml` 的配置文件。使用 `asadmin` 工具可以在 `domain.xml` 文件中创建负载平衡器配置。配置了负载平衡环境后，请将其导出到文件中：

1. 使用 `asadmin` 命令 `export-http-lb-config` 导出 `loadbalancer.xml` 文件。

   导出用于特定负载平衡器配置的 `loadbalancer.xml` 文件。您可以指定路径和其他文件名。如果不指定文件名，则该文件将被命名为 `loadbalancer.xml.`*load_balancer_config_name*。如果不指定路径，则将在 *application_server_install_dir*`/domains/`*domain_name*`/generated` 目录中创建该文件。

   要在 Windows 上指定路径，请用引号将路径引起来。例如，`"c:\sun\AppServer\loadbalancer.xml"`。

2. 将已导出的负载平衡器配置文件复制到 Web 服务器的配置目录。

   例如，对于 Sun Java System Web Server，该位置可能为 *web_server_root*/config。

   Web 服务器配置目录中的负载平衡器配置文件名必须为 loadbalancer.xml。如果您的文件使用其他名称 （例如 loadbalancer.xml.*load_balancer_config_name*），则必须进行重命名。

更改 HTTP 负载平衡器配置

如果您通过创建或删除对服务器的引用、部署新的应用程序、启用或禁用服务器或应用程序等方法来更改 HTTP 负载平衡器配置，请再次导出负载平衡器配置文件并将该文件复制到 Web 服务器的 config 目录。有关更多信息，请参见第 74 页上的“导出负载平衡器配置文件”。

负载平衡器插件将根据在负载平衡器配置中指定的重新装入时间间隔定期检查已更新的配置。在指定的时间值后，如果负载平衡器发现新的配置文件，它将开始使用该配置。

启用动态重新配置

启用动态重新配置后，负载平衡器插件将定期检查已更新的配置。要启用动态重新配置，请执行以下步骤：

- 要在创建负载平衡器配置时启用动态重新配置，请在运行 asadmin create-http-lb-config 时使用 --reloadinterval 选项。

  此选项用于设置检查负载平衡器配置文件 loadbalancer.xml 的更改的时间间隔。 0 值禁用重新装入。默认情况下，将启用动态重新装入，并且时间间隔设置为 60 秒。

- 要在禁用动态重新装入后将其启用，或更改重新装入间隔，请使用 asadmin set 命令。

在更改这些设置后，请再次导出负载平衡器配置文件并将其复制到 Web 服务器的 config 目录。

如果启用了先前已禁用的动态重新配置，您还必须重新启动 Web 服务器。

注

- 如果负载平衡器在尝试进行自我重新配置时遇到硬盘读取错误，它将使用内存中的当前配置。负载平衡器还确保了在覆写现有配置之前，已修改的配置数据符合 DTD。

  遇到磁盘读取错误后，将在 Web 服务器的错误日志文件中记录一则警告消息。

  Sun Java System Web Server 的错误日志位于：
  web_server_install_dir/*webserver_instance*/logs/。

禁用 （停止）服务器实例或群集

在因任何原因停止应用程序服务器之前，您希望实例完成正在处理的请求。正常禁用服务器实例或群集的进程称为停止。

负载平衡器使用以下策略来停止应用程序服务器实例：

- 如果已禁用某个实例 （独立实例或群集的一部分），并且超时尚未到期，粘性请求将继续发送到该实例。但是，新请求将不会发送到已禁用的实例。
- 超时到期后，该实例将被禁用。从负载平衡器到该实例的所有打开的连接将被关闭。即使并非所有粘连至该实例的会话均已失效，负载平衡器也不会将任何请求发送到该实例。负载平衡器会将粘性请求故障转移到另一个正常实例上。

要禁用某个服务器实例或群集，请执行以下步骤：

1. 运行 `asadmin disable-http-lb-server`，设置超时值 （以分钟为单位）。
2. 使用 `asadmin export-http-lb-config` 导出负载平衡器配置文件。
3. 将导出的配置复制到 Web 服务器的 `config` 目录。
4. 停止该服务器实例或群集。

禁用 （停止）应用程序

在取消部署 Web 应用程序之前，您希望该应用程序完成正在处理的请求。正常禁用应用程序的进程称为停止。

负载平衡器使用以下策略来停止应用程序：

- 如果禁用了某个应用程序，并且超时尚未到期，负载平衡器将不会转发对已禁用的应用程序的新请求。这些请求将被返回到 Web 服务器。粘性请求将被继续转发，直至超时到期。

- 超时到期后，该应用程序将被禁用。负载平衡器将不接受对该应用程序的任何请求 （包括粘性请求）。

当您从负载平衡器引用的每个服务器实例或群集中禁用应用程序时，则在再次启用该应用程序之前，已禁用的应用程序的用户将遭受服务损失。

如果您从一个服务器实例或群集中禁用应用程序而使该应用程序在其他服务器实例或群集中保持启用状态，则用户仍可访问该应用程序。

要禁用某个应用程序，请执行以下步骤：

1. 运行 `asadmin disable-http-lb-application`，指定超时值 （以分钟为单位）、要禁用的应用程序的名称以及要禁用该应用程序的目标群集或实例。
2. 使用 `asadmin export-http-lb-config` 导出负载平衡器配置文件。
3. 将导出的配置复制到 Web 服务器的 `config` 目录。

配置 HTTP 和 HTTPS 会话故障转移

如果 HTTP/HTTPS 会话所连接的原始应用程序服务器实例变为不可用，负载平衡器插件会将这些会话故障转移到其他应用程序服务器实例上。本节介绍了如何配置负载平衡器插件以启用 HTTP/HTTPS 路由选择和会话故障转移。

有关配置 HTTP 会话持久性的信息，请参见 “配置可用性和会话持久性”。

本节包括以下主题：

关于 HTTPS 路由选择

所有收到的请求 （无论是 HTTP 请求还是 HTTPS 请求）都是通过负载平衡器插件路由到应用程序服务器实例。但是，如果启用了 HTTPS 路由选择，则负载平衡器插件将仅把 HTTPS 请求转发给使用 HTTPS 端口的应用程序服务器。请注意，HTTPS 路由选择是针对新请求和粘性请求而执行的。

如果收到了 HTTPS 请求且没有正在进行的会话，负载平衡器插件将选择使用已配置的 HTTPS 端口的可用应用程序服务器实例，并将请求转发到该实例。

在正在进行的 HTTP 会话中，如果收到对同一个会话的新 HTTPS 请求，则将使用在 HTTP 会话期间保存的会话和粘性信息来路由 HTTPS 请求。新的 HTTPS 请求将被路由到处理上一个 HTTP 请求的同一服务器上，但是，是在 HTTPS 端口上进行。

配置 HTTPS 路由选择

create-http-lb-config 命令的 httpsrouting 选项用于控制是为正在参与负载平衡的所有应用程序服务器打开还是关闭 HTTPS 路由选择。如果此选项设置为 False，则所有 HTTP 和 HTTPS 请求都将作为 HTTP 请求进行转发。创建新的负载平衡器配置时，请将此选项设置为 True，或者以后使用 asadmin set 命令进行更改。

注

- 要使用 HTTPS 路由选择，必须配置一个或多个 HTTPS 侦听器。
- 如果 https-routing 被设置为 true，而新请求或粘性请求传入到没有正常 HTTPS 侦听器的群集中，该请求将生成一个错误。

有关负载平衡 HTTP/HTTPS 请求的已知问题

以下内容讨论了负载平衡器中有关 HTTP/HTTPS 请求处理的限制。

- 如果某个会话使用 HTTP 和 HTTPS 请求的组合，则第一个请求必须是 HTTP 请求。如果第一个请求是 HTTPS 请求，它后面将不能跟 HTTP 请求。这是因为与 HTTPS 会话关联的 Cookie 不是由浏览器返回的。浏览器将两个不同的协议解释为两个不同的服务器，并启动新的会话。

  仅当 httpsrouting 被设置为 true 时，此限制才有效。

- 如果某个会话具有 HTTP 和 HTTPS 请求的组合，则必须将应用程序服务器实例配置为同时具有 HTTP 和 HTTPS 侦听器。

  仅当 httpsrouting 被设置为 true 时，此限制才有效。

- 如果某个会话具有 HTTP 和 HTTPS 请求的组合，则必须将应用程序服务器实例配置为具有使用标准端口号 （即， HTTP 为 80， HTTPS 为 443）的 HTTP 和 HTTPS 侦听器。

  不管为 httpsrouting 设置了何值，此限制都有效。

配置幂等 URL

要增强已部署应用程序的可用性，请在负载平衡器所服务的所有应用程序服务器实例上将环境配置为重试失败的幂等 HTTP 请求。此选项用于只读请求 （例如，重试搜索请求）。

幂等请求是一种在重试时不会在应用程序中造成任何更改或不一致的请求。在 HTTP 中，某些方法 （例如 GET）是幂等的，而其他方法 （例如 POST）则不是。重试幂等 URL 不能导致服务器或数据库中的值发生更改。唯一的区别在于用户收到的响应会有所不同。

幂等请求的示例包括搜索引擎查询和数据库查询。基本原则是重试不会导致数据的更新或修改。

请在 sun-web.xml 文件中配置幂等 URL。当您导出负载平衡器配置时，幂等 URL 信息将自动被添加到 loadbalancer.xml 文件中。

有关配置幂等 URL 的更多信息，请参见 Developer's Guide。

配置 HTML 错误页面

您可以将自己的错误页面或某个错误页面的 URL 指定为显示给最终用户。指定错误页面将会覆盖为错误报告配置的所有其他机制。

请在 sun-web.xml 文件中配置 HTML 错误页面。在导出负载平衡器配置时，系统会自动将 HTML 错误页面信息从 sun-web.xml 文件添加到 loadbalancer.xml 文件中。

有关配置 HTML 错误页面的更多信息，请参见 Developer's Guide。

监视 HTTP 负载平衡器插件

配置日志消息

负载平衡器插件使用 Web 服务器的日志机制来生成日志消息。 Application Server 上的默认日志级别被设置为 Sun Java System Web Server (INFO)、 Apache Web Server (WARN) 和 Microsoft IIS (INFO) 上的默认日志级别。应用程序服务器日志级别（FINE、 FINER 和 FINEST）映射到 Web 服务器上的 DEBUG 级别。

这些日志消息将被写入 Web 服务器日志文件，其形式为可使用脚本进行解析或可被导入电子表格以计算所需的衡量标准的原始数据。

日志消息类型

负载平衡器插件生成以下三组不同的日志消息：

- 负载平衡器配置器日志消息
- 请求分发和运行时日志消息
- 配置器错误消息

负载平衡器配置器日志消息

使用幂等 URL 和错误页面设置时，将记录这些消息。

幂等 URL 模式配置的输出包含以下信息：

- 当日志级别被设置为详细时：

  ```
  CONFxxxx: IdempotentUrlPattern configured <url-pattern>
  <no-of-retries> for web-module : <web-module>
  ```

- 当日志级别被设置为严重时：

  ```
  CONFxxxx: Duplicate entry of Idempotent URL element <url-pattern>
  for webModule <web-module> in loadbalancer.xml."
  ```

- 当日志级别被设置为警告时：

  ```
  CONFxxxx: Invalid IdempotentUrlPatternData <url-pattern> for
  web-module <web-module>
  ```

错误页面 URL 配置的输出包含以下信息 （日志级别设置为警告）：

```
CONFxxxx: Invalid error-url for web-module <web-module>
```

请求分发和运行时日志消息

在对请求进行负载平衡和分发时，将生成这些日志消息。

- 每个方法开始的标准日志的输出均包含以下信息 （日志级别设置为详细）：

  ```
  ROUTxxxx: Executing Router method <method_name>
  ```

- 每个方法开始的路由器日志的输出均包含以下信息 （日志级别设置为信息）：

  ```
  ROUTxxxx: Successfully Selected another ServerInstance for
  idempotent request <Request-URL>
  ```

- 运行时日志的输出包含以下信息 （日志级别设置为信息）：

  ```
  RNTMxxxx: Retrying Idempotent <GET/POST/HEAD> Request <Request-URL>
  ```

配置器错误消息

如果存在配置问题 （例如，缺少引用的自定义错误页面），将显示这些错误消息。

- 日志级别设置为信息：

  ROUTxxxx: Non Idempotent Request <Request-URL> cannot be retried

  ```
  Example :: ROUTxxxx: Non Idempotent Request
  http://sun.com/addToDB?x=11&abc=2 cannot be retried
  ```

- 日志级别设置为详细：

  RNTMxxxx: Invalid / Missing Custom error-url / page:<error-url> for web-module:<web-module>

  ```
  Example :: RNTMxxxx: Invalid / Missing Custom error-url /
  page:myerror1xyz for web-module: test
  ```

配置监视

执行以下步骤可以打开负载平衡器插件日志消息：

1. 设置 Web 服务器中的日志选项。

   a. 从 Sun Java System Web Server 的管理控制台中，转到 “Magnus 编辑器”选项卡。

      将 `Log Verbose` 选项设置为 `On`。

   b. 对于 Apache Web Server，请将日志级别设置为 `DEBUG`。

   c. 对于 Microsoft IIS，请在 `sun-passthrough.properties` 文件中将日志级别设置为 `FINE`。

2. 将负载平衡器配置的 `monitor` 选项设置为 true。

   使用 `asadmin create-http-lb-config` 命令在最初创建负载平衡器配置时将监视设置为 true，也可以在以后使用 `asadmin set` 命令将其设置为 true。默认情况下，监视处于禁用状态。

负载平衡器插件记录以下信息：

- 每个请求的请求开始/停止信息。
- 当请求从异常实例故障转移到正常实例时的故障转移请求信息。
- 每个运行状况检查周期结束时的异常实例列表。

注 在负载平衡器插件上启用日志记录后，如果将 Web 服务器日志级别设置为 DEBUG 或设置为打印详细消息，负载平衡器会将 HTTP 会话 ID 写入 Web 服务器日志文件中。因此，如果托管负载平衡器插件的 Web 服务器位于 DMZ 中，请不要在生产环境中使用 DEBUG 或类似的日志级别。

如果必须使用 DEBUG 日志级别，请在 loadbalancer.xml 中将 require-monitor-data 属性设置为 false，以关闭负载平衡器日志记录。

了解监视消息

负载平衡器插件日志消息的格式如下所示：

- HTTP 请求的开头处包含以下信息：

  ```
  RequestStart Sticky(New) <req-id> <time-stamp> <URL>
  ```

  时间戳值是从 1970 年 1 月 1 日开始的毫秒数。例如：

  ```
  RequestStart New 123456 602983
  http://austen.sun.com/Webapps-simple/servlet/Example1
  ```

- HTTP 请求的结尾处包含 RequestExit 消息，如下所示：

  ```
  RequestExit Sticky(New) <req-id> <time-stamp> <URL> <listener-id>
  <response-time> Failure-<reason for error>(incase of a failure)
  ```

  例如：

  ```
  RequestExit New 123456 603001
  http://austen.sun.com/Webapps-simple/servlet/Example1
  http://austen:2222 18
  ```

  注 在 RequestExit 消息中， <response-time> 表示从负载平衡器插件方面，请求的往返总时间 （以毫秒为单位）。

- 异常实例列表，如下所示：

  ```
  UnhealthyInstances <cluster-id> <time-stamp> <listener-id>,
  <listener-id>...
  ```

  例如：

  ```
  UnhealthyInstances cluster1 701923 http://austen:2210, http://austen:3010
  ```

- 故障转移请求列表，如下所示：

  ```
  FailedoverRequest <req-id> <time-stamp> <URL> <session-id>
  <failed-over-listener-id> <unhealthy-listener-id>
  ```

  例如：

  ```
  FailedoverRequest 239496 705623
  http://austen.sun.com/Apps/servlet/SessionTest 16dfdac3c7e80a40
  http://austen:4044 http://austen:4045
  ```

升级应用程序

关于滚动升级

要升级应用程序而不使用户遭受任何服务损失，请每次在一个服务器或群集上升级应用程序。群集以透明方式维护一个混合版本环境，用户不会察觉到正在进行升级。这种升级称为滚动升级。

只有在旧版本和新版本的应用程序兼容并且能够同时运行的情况下，才能执行滚动升级。会话信息必须兼容。可以在单个独立群集或在多个群集中执行混合模式的滚动升级。

如果应用程序发生重大更改（例如，对数据库模式的更改），则无法执行混合模式环境中的滚动升级。在这种情况下，升级时请关闭此应用程序。取消部署此应用程序，然后使用相同的名称重新部署已升级的应用程序。

在单个独立群集中进行升级

要在单个独立群集 （即不与任何其他群集共享配置的群集）中升级应用程序，请执行以下步骤：

1. 保存旧版本的应用程序或备份域。

   要备份域，请使用 `asadmin backup-domain` 命令。

2. 关闭群集的动态重新配置 （如果已启用）。

   通过管理控制台：

   a. 展开 “配置”节点。

   b. 单击群集配置的名称。

   c. 在 “配置系统属性”页面中，取消选中 “已启用动态重新配置”框。

   d. 单击 “保存”

   等效的 `asadmin` 命令为 `asadmin set`。语法为：

   `asadmin set --user` *user* `--passwordfile` *password_file*
   cluster_name`-config.dynamic-reconfiguration-enabled=false`

3. 将已升级的应用程序重新部署到目标 `domain` 中。如果使用管理控制台进行重新部署，域将自动成为目标。由于已禁用动态重新配置，因此旧应用程序将继续在群集上运行。

4. 使用 `asadmin enable-http-lb-application` 为实例启用已重新部署的应用程序。

5. 使用 `asadmin disable-http-lb-server` 禁用一个服务器实例。

6. 使用 `asadmin export-http-lb-config` 导出负载平衡器配置文件。

7. 将已导出的配置文件复制到 Web 服务器实例的配置目录。例如，对于 Sun Java System Web Server，其位置为
   web_server_install_dir`/https-`*host-name*`/config/loadbalancer.xml`

8. 请等待，直至超时到期。监视负载平衡器的日志文件以确保实例已脱机。

9. 在群集中的其他实例仍处于运行状态的情况下，重新启动已禁用的服务器实例。重新启动操作将使服务器与域同步，并更新应用程序。

10. 测试重新启动的服务器上的应用程序，以确保应用程序运行正常。

11. 使用 asadmin enable-http-lb-server 启用服务器实例。
12. 使用 asadmin export-http-lb-config 导出负载平衡器配置文件。
13. 将配置文件复制到 Web 服务器的配置目录。
14. 对群集中的每个实例重复步骤 5 至步骤 13。
15. 当所有服务器实例都包含新的应用程序并且处于运行状态时，请再次为群集启用动态重新配置。

在两个群集中进行升级

1. 保存旧版本的应用程序或备份域。

   要备份域，请使用 asadmin backup-domain 命令。

2. 关闭两个群集的动态重新配置 （如果已启用）。

   通过管理控制台：

   a. 展开 “配置”节点。
   b. 单击其中一个群集配置的名称。
   c. 在 “配置系统属性”页面中，取消选中 “已启用动态重新配置”框。
   d. 单击 “保存”
   e. 对第二个群集重复此过程。

   等效的 asadmin 命令为 asadmin set。语法为：

   asadmin set --user *user* --passwordfile *password_file* *cluster_name*-config.dynamic-reconfiguration-enabled=false

3. 将已升级的应用程序重新部署到目标 domain 中。如果使用管理控制台进行重新部署，域将自动成为目标。由于已禁用动态重新配置，因此旧应用程序将继续在群集上运行。
4. 使用 asadmin enable-http-lb-application 为群集启用已重新部署的应用程序。
5. 使用 asadmin disable-http-lb-server 从负载平衡器上禁用其中一个群集。
6. 使用 asadmin export-http-lb-config 导出负载平衡器配置文件。

7. 将已导出的配置文件复制到 Web 服务器实例的配置目录。例如，对于 Sun Java System Web Server，其位置为 *web_server_install_dir*/https-*host-name*/config/loadbalancer.xml
8. 请等待，直至超时到期。监视负载平衡器的日志文件以确保群集已脱机。
9. 在另一个群集仍处于运行状态的情况下，重新启动已禁用的群集。重新启动操作将导致群集与域同步，并更新应用程序。
10. 测试重新启动的群集上的应用程序，以确保应用程序运行正常。
11. 使用 `asadmin enable-http-lb-server` 启用群集。
12. 使用 `asadmin export-http-lb-config` 导出负载平衡器配置文件。
13. 将配置文件复制到 Web 服务器的配置目录。
14. 对另一个群集重复步骤 5 至步骤 13。
15. 当所有服务器实例都包含新的应用程序并且处于运行状态时，请再次为两个群集启用动态重新配置。

关于 RMI-IIOP 负载平衡和故障转移

RMI-IIOP 负载平衡和故障转移的要求

Sun Java™ System Application Server 通过 RMI-IIOP 提供远程 EJB 引用和名称服务对象的高可用性。在使用这些功能之前，环境必须满足以下要求：

- Sun Java System 已安装 Application Server Enterprise Edition。
- 存在一个至少包括两个应用程序服务器实例的群集。

  有关群集的更多信息，请参见 “配置群集”。
- J2EE 应用程序已被部署到所有将参与负载平衡的应用程序服务器实例和群集。

- 已启用 RMI-IIOP 客户机应用程序以用于负载平衡。

  系统支持以下两种 RMI/IIOP 客户机的负载平衡：

  - 在应用程序客户机容器 (ACC) 中运行且访问部署在 Application Server 实例上的 EJB 的 Java 应用程序。
  - 不在 ACC 中运行、但访问部署在 Application Server 实例上的 EJB 的 Java 应用程序。

  启用基于 RMI-IIOP 的应用程序所需的配置设置取决于客户机的类型。有关配置用于负载平衡的 RMI-IIOP 客户机应用程序的更多信息，请参见 《Sun Java System Application Server Developer's Guide》。

  有关 RMI-IIOP 故障转移和负载平衡的其他信息，请参见 《Sun Java System Application Server High Availability Administration Guide》。

注 不支持通过 SSL 进行 RMI-IIOP 故障转移。

RMI-IIOP 负载平衡和故障转移算法

Sun Java System Application Server 采用了随机算法和 Round Robin 算法，用于 RMI-IIOP 路径上的远程 EJB 引用和名称服务对象的负载平衡。

在 RMI-IIOP 客户机首次创建新的 `InitialContext` 对象时，可用的 Application Server IIOP 端点的列表对于该客户机是随机的。对于该 `InitialContext` 对象，负载平衡器会将查找请求和其他 `InitialContext` 操作定向至列表中的第一个端点。如果第一个端点不可用，则使用列表中的第二个端点，依此类推。

随后每次客户机创建新的 `InitialContext` 对象时，将轮转端点列表，从而将不同的 IIOP 端点用于 `InitialContext` 操作。

当您从通过 `InitialContext` 对象获得的引用中获取或创建 Bean 时，将在服务于指定给 `InitialContext` 对象的 IIOP 端点的 Application Server 实例上创建这些 Bean。对这些 Bean 的引用包括群集中的所有 Application Server 实例的 IIOP 端点地址。

主端点是与用于查找或创建 Bean 的 `InitialContext` 端点相对应的 Bean 端点。群集中的其他 IIOP 端点将被指定为**备用端点**。如果 Bean 的主端点不可用，则该 Bean 上的其他请求将故障转移到其中一个备用端点。

RMI-IIOP 样例应用程序

以下目录包含用于演示通过和不通过 ACC 使用 RMI-IIOP 故障转移的样例应用程序。

install_dir/samples/ee-samples/sfsbfailover

有关通过和不通过 ACC 来运行应用程序的说明，请参见该样例附带的 index.html 文件。 ee-samples 目录还包含有关设置运行样例的环境的信息。

第 4 章

配置节点代理

本章介绍了 Application Server 中的节点代理。它包含以下几节：

关于节点代理

节点代理

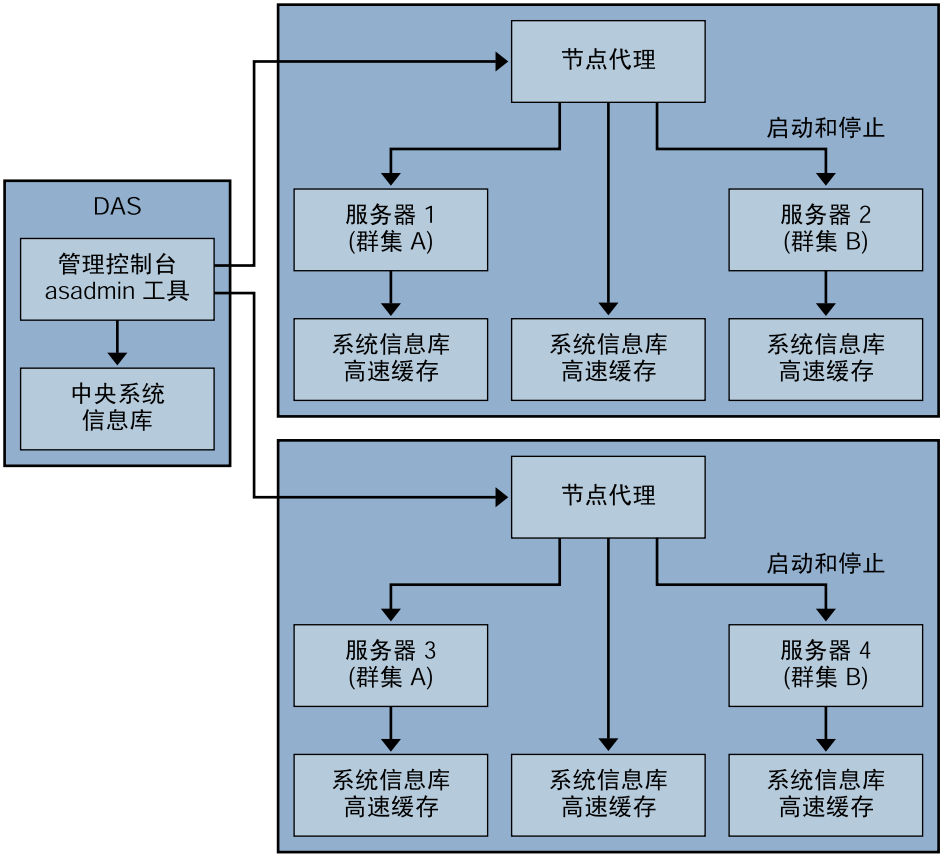
节点代理是托管服务器实例的每台计算机上都需要的轻量代理，包括托管域管理服务器 (DAS) 的计算机。节点代理可以：

- 按照域管理服务器的指示启动、停止、创建和删除服务器实例。
- 重新启动有故障的服务器实例。

- 提供有故障的服务器的日志文件视图。
- 使每个服务器实例的本地配置系统信息库与域管理服务器的中心系统信息库同步。每个本地系统信息库只包含与该服务器实例或节点代理相关的信息。

下图说明了节点代理的整个体系结构：

图 4-1 节点代理体系结构



自动创建的节点代理

安装 Application Server 时，默认情况下将使用该计算机的主机名创建节点代理。必须先在本地计算机上手动启动节点代理之后，该节点代理才能运行。

节点代理和服务器实例管理

即使未运行节点代理，您也可以创建和删除服务器实例。但是，节点代理必须处于运行状态，您才能用它来启动和停止服务器实例。

如果停止节点代理，则该节点代理管理的服务器实例也将被停止。

其他节点代理

一个节点代理服务一个域。如果一台计算机托管在多个域中运行的实例，则该计算机必须运行多个节点代理。

节点代理占位符

您可以使用节点代理占位符创建和删除不具有现有节点代理的服务器实例。占位符是在节点代理的本地系统中创建节点代理本身之前，在域管理服务器 (DAS) 上创建的节点代理配置。

注 当您创建占位符节点代理之后，即可使用该占位符节点代理在域中创建实例。但是，在启动实例之前，您必须先使用 `asadmin` 命令在实例将要驻留的计算机上从本地创建并启动实际的节点代理。请参见第 101 页上的 “创建节点代理”和第 102 页上的 “启动节点代理”了解更多信息。

部署节点代理

您可以使用以下两种方式之一来配置和部署节点代理：

- 联机部署。如果在配置域之前，您已知道该域的拓扑并且具有用于该域的硬件，则请使用联机部署
- 脱机部署。如果是在设置整个环境之前配置域和服务器实例，则请使用脱机部署

部署节点代理之前

部署节点代理之前，请执行以下步骤：

1. 安装域管理服务器。
2. 启动域管理服务器。

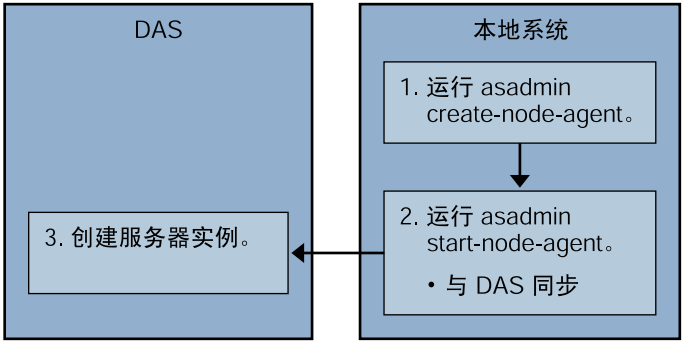
启动并运行域管理服务器后，便可以开始进行联机或脱机部署。

联机部署

如果开始配置域之前已经知道该域的拓扑并且具有用于该域的硬件，则可以通过联机部署来配置域。

下图总结了节点代理的联机部署：

图 4-2 联机节点代理部署



要配置联机部署，请执行以下步骤：

1. 在将要托管服务器实例的每台计算机上安装节点代理。

   使用安装程序或 `asadmin` 命令 `create-node-agent`。如果某台计算机需要多个节点代理，请使用 `asadmin` 命令 `create-node-agent` 来创建这些节点代理。

   详细信息，请参见第 101 页上的 “创建节点代理”。

2. 使用 `asadmin` 命令 `start-node-agent` 启动节点代理。

   启动之后，节点代理将与域管理服务器进行通信 (DAS)。当节点代理到达 DAS 时，将在 DAS 上创建该节点代理的配置。具备了配置之后，即可在管理控制台中查看该节点代理。

   详细信息，请参见第 102 页上的 “启动节点代理”。

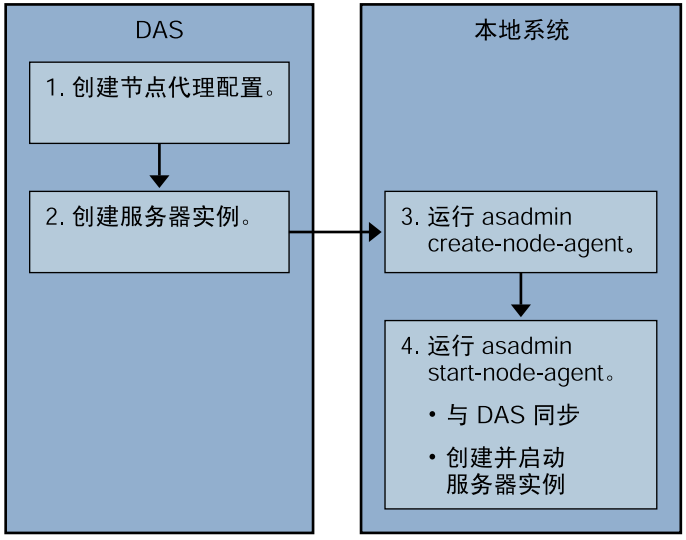
3. 配置域，包括创建服务器实例和群集以及部署应用程序。

脱机部署

脱机方法使您可以在配置各个本地计算机之前轻松地定义和重新安排域中的项。

下图总结了脱机部署的步骤。

图 4-3 脱机节点代理部署



在本地计算机中设置节点代理之前要配置域和服务器实例 （脱机配置），请执行以下步骤：

1. 在域管理服务器中创建占位符节点代理。

   详细信息，请参见第 98 页上的 “创建节点代理占位符”。

2. 创建服务器实例和群集并部署应用程序。

3. 在将要托管服务器实例的每台计算机上安装节点代理。

   使用安装程序或 `asadmin` 命令 `create-node-agent`。节点代理的名称必须与先前创建的占位符节点代理的名称相同。

   详细信息，请参见第 101 页上的 “创建节点代理”。

4. 使用 asadmin 命令 start-node-agent 启动节点代理。

   节点代理启动之后，它将绑定到域管理服务器并创建先前已与该节点代理相关联的所有服务器实例。

   详细信息，请参见第 102 页上的 “启动节点代理”。

节点代理和域管理服务器同步

由于配置数据既存储在域管理服务器的系统信息库 （中心系统信息库）中，又高速缓存在节点代理的本地计算机中，因此这两者必须同步。

节点代理同步

第一次启动节点代理时，该节点代理将向域管理服务器 (DAS) 发送一个请求，以获得中心系统信息库中的最新信息。当节点代理成功地与 DAS 取得联系并获得配置信息时，该节点代理即**绑定**到该 DAS。

如果您在 DAS 上创建了占位符节点代理，则第一次启动节点代理时，该节点代理将从 DAS 的中心系统信息库中获取其配置。

初始启动节点代理过程中，如果由于没有运行 DAS 而使该节点代理无法到达 DAS，则该节点代理将停止并保持**未绑定**状态。

如果在域中更改了节点代理的配置，这些更改将在节点代理运行时自动与本地计算机中的节点代理进行通信。

如果删除了 DAS 中的一个节点代理配置，则下次该节点代理进行同步时将停止它自身并将自身标记为待删除状态。使用本地 asadmin 命令 delete-node-agent 手动删除该节点代理。

服务器实例同步

如果使用管理控制台或 asadmin 工具显式启动了服务器实例，则该服务器实例将与中心系统信息库同步。如果此同步失败，则服务器实例不会启动。

如果节点代理未通过向管理控制台或 asadmin 工具发送显式请求来启动服务器实例，将不同步该服务器实例的系统信息库高速缓存。该服务器实例将以存储在其高速缓存中的配置运行。

同步大型应用程序

当环境中有大型应用程序要进行同步或者可用内存受到限制时，您可以调整 JVM 选项以限制内存的使用。这种调整将减少收到内存不足错误的可能性。实例同步 JVM 使用的是默认设置，但您可以配置 JVM 选项来更改这些设置。

可以使用 INSTANCE-SYNC-JVM-OPTIONS 属性设置 JVM 选项。设置属性的命令为：

```
asadmin set domain.node-agent.node_agent_name.property.INSTANCE-SYNC-JVM-OPTIONS="JVM_options"
```

例如：

```
asadmin set domain.node-agent.node0.property.INSTANCE-SYNC-JVM-OPTIONS="-Xmx32m -Xss2m"
```

在此示例中，节点代理为 node0， JVM 选项为 -Xmx32m -Xss2m。

有关 JVM 选项的更多信息，请参见：

http://java.sun.com/docs/hotspot/VMOptions.html

注 更改 INSTANCE-SYNC-JVM-OPTIONS 属性后，请重新启动节点代理，因为在节点代理的配置中添加或更改了某个属性时节点代理并不会自动同步。

查看节点代理日志

每个节点代理都有自己的日志文件。如果使用节点代理时遇到问题，请参见位于以下位置的日志文件：

node_agent_dir/*node_agent_name*/agent/logs/server.log。

有时，节点代理日志会指示您查看服务器的日志以获得关于所出现的问题的详细消息。

服务器日志位于：

node_agent_dir/*node_agent_name*/*server_name*/logs/server.log

node_agent_dir 的默认位置为 *install_dir*/nodeagents。

可以通过管理控制台和 asadmin 工具执行的任务

对于节点代理，有些任务必须从运行该节点代理的系统本地执行，而其他任务则可以在域管理服务器中执行。需要本地执行的任务只能在节点代理驻留的计算机中通过运行 asadmin 工具来执行。在域管理服务器中运行的任务则可以通过管理控制台和 asadmin 工具来执行。

下表总结了这些任务以及执行这些任务的方式：

表 4-1 可以通过管理控制台和 asadmin 命令执行的任务

任务	管理控制台	asadmin 命令
在域管理服务器中创建节点代理占位符/配置	“创建节点代理占位符”页面	create-node-agent-config
创建节点代理	不可用	create-node-agent
启动节点代理	不可用	start-node-agent
停止节点代理	不可用	stop-node agent
从域管理服务器中删除节点代理配置	“节点代理”页面	delete-node-agent-config
从本地计算机中删除节点代理	不可用	delete-node-agent
编辑节点代理配置	“节点代理”页面	set
列出节点代理	“节点代理”页面	list-node-agents

用于节点代理的管理控制台任务

- 查看常规节点代理信息
- 创建节点代理占位符
- 删除节点代理配置
- 编辑节点代理配置
- 编辑节点代理区域
- 编辑节点代理的 JMX 侦听器

查看常规节点代理信息

创建节点代理或节点代理占位符后，便可以查看其设置：

1. 在树组件中，选择 “节点代理”节点。
2. 单击一个节点代理的名称。

   如果节点代理已经存在但并未在此处显示，请在节点代理的主机上使用 `asadmin start-node-agent` 来启动该节点代理。详细信息，请参见第 102 页上的 “启动节点代理”。
3. 检查节点代理的主机名。

   如果主机名为 “未知主机”，则该节点代理没有同域管理服务器 (DAS) 进行初始联系。
4. 检查节点代理的状态。

   正在运行。已正确创建节点代理，并且当前该节点代理正在运行。

   未运行。存在以下情形之一：

   - 已在本地计算机中创建了节点代理，但从未启动过该节点代理。
   - 以前启动过节点代理，但现在已停止。

   等待会合。节点代理是从未在本地计算机中创建的占位符。

   有关创建和启动节点代理的更多信息，请参见第 101 页上的 “创建节点代理”和第 102 页上的 “启动节点代理”。
5. 选择启动节点代理时是否启动实例。

   选择 “是”，将在启动节点代理时自动启动与该节点代理关联的服务器实例。选择 “否”以手动启动这些实例。
6. 确定节点代理是否已同域管理服务器进行了联系。

   如果节点代理从未与域管理服务器进行过联系，则该节点代理从未被成功启动过。
7. 管理与节点代理关联的服务器实例。

   如果节点代理正在运行，则通过单击实例名称旁边的复选框并单击 “启动”或 “停止”来启动或停止实例。

创建节点代理占位符

由于必须从托管节点代理的计算机本地创建节点代理，因此您只能通过管理控制台为节点代理创建占位符。此占位符是尚不存在的节点代理的节点代理配置。

创建一个占位符之后，请在托管节点代理的计算机中使用 asadmin 命令 create-node-agent 完成节点代理的创建。有关更多信息，请参见第 101 页上的“创建节点代理”。

1. 在树组件中，选择“节点代理”节点。
2. 在“节点代理”页面中，单击“新建”。
3. 在“当前节点代理占位符”页面中为新节点代理输入一个名称。

   在域中的所有节点代理名称、服务器实例名称、群集名称和配置名称中，此名称必须是唯一的。
4. 单击“确定”。

   新节点代理的占位符将列出在“节点代理”页面中。

等效的 asadmin 命令为：create-node-agent-config。

删除节点代理配置

通过管理控制台，您只能删除域中的节点代理配置，而不能删除实际的节点代理。要删除节点代理本身，请在该节点代理的本地计算机中运行 asadmin 命令 delete-node-agent。有关更多信息，请参见第 103 页上的“删除节点代理”。

删除节点代理配置之前，必须停止该节点代理并且该节点代理不能有任何关联的实例。要停止节点代理，请使用 asadmin 命令 stop-node-agent。详细信息，请参见第 102 页上的“停止节点代理”。

1. 在树组件中，选择“节点代理”节点。
2. 在“节点代理”页面中，选中要删除的节点代理旁边的复选框。
3. 单击“删除”。

等效的 asadmin 命令为：delete-node-agent-config。

编辑节点代理配置

要编辑节点代理配置，请执行以下步骤：

1. 在树组件中，展开 “节点代理”节点。
2. 选择要编辑的节点代理配置。
3. 在 “节点代理常规信息”页面中，选择启动代理时是否启动代理的服务器实例。您还可以从此页面手动启动和停止实例。

如果此配置用于占位符节点代理，则使用 `asadmin create-node-agent` 创建实际的节点代理时，实际的节点代理将使用该配置。有关创建节点代理的信息，请参见第 101 页上的 “创建节点代理”。

如果此配置是现有节点代理的配置，则将自动同步此节点代理配置信息。

编辑节点代理区域

为连接到节点代理的用户设置验证区域。只有管理用户才能访问节点代理。

1. 在树组件中，展开 “节点代理”节点。
2. 选择要编辑的节点代理配置。
3. 单击 “验证区域”选项卡。
4. 在 “节点代理编辑区域”页面中，输入一个区域。

   默认区域为 `admin-realm`，它是您在创建节点代理时创建的。要使用其他区域，请用新区域替换由域控制的**所有**组件中的区域，否则这些组件将无法正常通信。
5. 在 “类名”字段中，指定实现该区域的 Java 类。
6. 添加需要的所有属性。

   验证区域需要特定于提供商的属性，这些属性会因特定实现的需求而有所不同。

编辑节点代理的 JMX 侦听器

节点代理使用 JMX 与域管理服务器进行通信。因此，该节点代理必须具有侦听 JMX 请求的端口和其他侦听器信息。

1. 在树组件中，展开 “节点代理”节点。
2. 选择要编辑的节点代理配置。
3. 单击 “JMX”选项卡。
4. 如果侦听器将侦听服务器的所有 IP 地址，请在 “地址”字段中键入 0.0.0.0 并使用唯一的端口值。否则，请键入此服务器的有效 IP 地址。
5. 在 “端口”字段中，键入节点代理的 JMX 连接器要侦听的端口值。如果 IP 地址为 0.0.0.0，则端口号必须唯一。
6. 在 “JMX 协议”字段中，键入 JMX 连接器支持的协议。

   默认值为 rmi_jrmp。
7. 单击 “接受所有地址”旁边的复选框以允许连接到所有 IP 地址。

   节点代理将侦听与网卡相关联的特定 IP 地址或侦听所有 IP 地址。如果接受所有地址，则会将值 `0.0.0.0` 放入 “侦听主机地址”属性中。
8. 在 “区域名称”字段中，键入为侦听器处理验证的区域的名称。

在此页面的 “安全性”部分中，您可以将侦听器配置为使用 SSL 安全性、 TLS 安全性或同时使用 SSL 和 TLS 安全性。

要设置安全侦听器，请执行以下操作：

1. 在 “安全性”字段中选中 “启用”框。

   默认情况下，启用 “安全性”。
2. 要使客户机在使用此侦听器时自行向服务器进行验证，请在 “客户机验证”字段中选中 “已启用”框。
3. 在 “证书昵称”字段中输入现有服务器密钥对和证书的名称。有关详细信息，请参见 “安全性”一章。

4. 在 "SSL3/TLS" 部分：
   a. 选取要在侦听器上启用的安全协议。必须选取 SSL3 或 TLS 或同时选取两者。
   b. 选取协议所使用的加密算法套件。要启用所有加密算法套件，请选取 “所有支持的加密算法套件”。
5. 单击 “保存”。

在 asadmin 工具中用于节点代理的任务

创建节点代理

要创建节点代理，请在运行该节点代理的计算机中本地运行 `asadmin` 命令 `create-node-agent`。

例如：

```
$ asadmin create-node-agent --host myhost --port 4849 ---user admin
nodeagent1
```

其中， `myhost` 是域管理服务器 (DAS) 主机名， `4849` 是 DAS 端口号， `admin` 是 DAS 用户， `nodeagent1` 是要创建的节点代理的名称。

节点代理的默认名称为创建节点代理时所在的主机的名称。

如果已创建节点代理占位符，请使用与节点代理占位符相同的名称来创建相应的节点代理。如果您尚未创建节点代理占位符，而 DAS 已启动并且可访问，则 `create-node-agent` 命令还将在 DAS 上创建节点代理配置 （占位符）。

有关命令语法的完整说明，请参见该命令的联机帮助。

注 在以下情况下，必须指定一个 DNS 可访问的主机名：

1. 域跨越了子网边界 （即，节点代理和域管理服务器 [DAS] 处于不同的域中，例如 sun.com 和 java.com）
2. 使用未在 DNS 中登记的主机名的 DHCP 计算机

创建域和节点代理时，可通过明确指明域和节点代理的主机名指定一个 DNS 可访问的主机名：

`create-domain --domainproperties domain.hostName=`*DAS-host-name*

`create-node-agent --host` *DAS-host-name* `--agentproperties remoteclientaddress=`*node-agent-host-name*

另外一种解决方案是更新特定于平台的 `hosts` 主机名/IP 解析文件，从而将主机名解析为正确的 IP 地址。但是，使用 DHCP 重新连接时，可能会为您分配不同的 IP 地址。在这种情况下，您必须更新每个服务器中的主机解析文件。

启动节点代理

必须先运行节点代理，然后它才能管理服务器实例。通过在节点代理所驻留的系统中以本地方式运行 `asadmin` 命令 `start-node-agent` 来启动节点代理。

例如：

```
$ asadmin start-node-agent --user admin nodeagent1
```

其中， `admin` 是管理员用户， `nodeagent1` 是要启动的节点代理。

有关命令语法的完整说明，请参见该命令的联机帮助。

停止节点代理

要停止正在运行的节点代理，请在该节点代理驻留的系统中运行 `asadmin` 命令 `stop-node-agent`。 `stop-node-agent` 命令将停止节点代理所管理的所有服务器实例。

例如：

```
$ asadmin stop-node-agent nodeagent1
```

其中的 `nodeagent1` 是节点代理。

有关命令语法的完整说明，请参见该命令的联机帮助。

删除节点代理

要删除节点代理文件，请在该节点代理驻留的系统中运行 asadmin 命令 delete-node-agent。

删除节点代理之前，必须先停止节点代理。您还可以删除从未启动过或者从未成功地与域管理服务器连接 （即尚未绑定）的节点代理。

例如：

```
$ asadmin delete-node-agent nodeagent1
```

其中的 nodeagent1 是节点代理。

有关命令语法的完整说明，请参见该命令的联机帮助。

删除节点代理时，您还必须使用管理控制台或 asadmin delete-node-agent-config 命令将该节点代理的配置从域管理服务器中删除。

第 5 章

部署应用程序

本章说明如何在 Application Server 上部署 （安装） J2EE 应用程序。本章包含以下几节：

- 关于部署
- 用于部署应用程序的管理控制台任务
- 用于列出、取消部署以及启用应用程序的管理控制台任务
- 适用于开发者的部署方法

关于部署

- 部署生命周期
- J2EE 归档文件的类型
- 命名约定

部署生命周期

安装 Application Server 并启动域之后，您可以部署 （安装） J2EE 应用程序和模块。在部署过程中和更改应用程序时，应用程序或模块可能会经过以下阶段：

1. 初始部署

   部署应用程序或模块之前，请启动域。

   将应用程序或模块部署 （安装）到特定的独立服务器实例或群集。由于应用程序和模块封装在归档文件中，因此在部署期间应指定归档文件名。默认情况下，部署到默认服务器实例 `server`。

   如果部署到服务器实例或群集，则应用程序或模块将存在于域的中心系统信息库中，并由部署到的所有目标群集或服务器实例引用。

   您还可以使用 `asadmin deploy` 命令 （而非管理控制台）将其部署到域。如果将应用程序或模块只部署到域，则应用程序或模块将存在于域的中心系统信息库中，但要在您添加引用之后才会有服务器实例或群集引用该应用程序或模块（如步骤 3 中所述）。

   部署是动态的：部署应用程序或模块后，无需重新启动服务器实例即可使用应用程序或模块。如果重新启动了服务器实例，所有已部署的应用程序和模块仍将处于部署状态并且可用。

2. 启用或禁用

   默认情况下，将启用已部署的应用程序或模块，这表示如果应用程序或模块已部署到可访问的服务器实例或群集，则可以运行它并且可以通过客户机对其进行访问。要禁止访问，请禁用应用程序或模块。在部署之后，已禁用的应用程序或模块并未从域中被卸载，而且可以很容易地将其启用。

3. 添加或删除已部署应用程序或模块的目标

   部署后，应用程序或模块将存在于中心系统信息库中，并可以被多个服务器实例和 / 或群集引用。最初，作为目标部署到的服务器实例或群集将引用应用程序或模块。

   在部署应用程序或模块后，要更改引用应用程序或模块的服务器实例和群集，请使用管理控制台更改应用程序或模块的目标，或使用 asadmin 工具更改应用程序引用。由于应用程序本身存储在中心系统信息库中，因此添加或删除目标将添加或删除不同目标上同一版本的应用程序。但是，可以在一个目标上启用而在另一个目标上禁用部署到多个目标的应用程序，因此即使应用程序被一个目标引用，也只有在该目标上启用它时用户才能对其进行使用。

4. 重新部署

   要替换已部署的应用程序或模块，请将其重新部署。重新部署将自动取消部署先前已部署的应用程序或模块，并用新的应用程序或模块对其进行替换。

   当通过管理控制台重新部署时，重新部署的应用程序或模块将部署到域中，并且所有对其进行引用的独立或群集服务器实例将自动接收新的版本 （如果已启用动态重新配置）。如果使用 `asadmin deploy` 命令来重新部署，请将 `domain` 指定为目标。

   对于生产环境，请使用滚动升级 （升级应用程序而不中断服务）。有关更多信息，请参见第 83 页上的 “关于滚动升级”。

5. 取消部署

   要卸载应用程序或模块，请取消部署应用程序或模块。

J2EE 归档文件的类型

软件供应商将应用程序或模块封装到了归档文件中。要部署应用程序或模块，请指定归档文件名。归档文件的内容和结构是按照 J2EE 平台的规范定义的。 J2EE 归档文件的类型包括：

- Web 应用程序归档文件 (WAR)：WAR 文件由 Servlet 和 JSP 等 Web 组件以及静态 HTML 页面、 JAR 文件、标记库和实用程序类组成。 WAR 文件名具有 `.war` 扩展名。
- EJB JAR：EJB JAR 文件包含一个或多个企业 Bean （用于 EJB 技术的组件）。 EJB JAR 文件还包括企业 Bean 所需的任何实用程序类。 EJB JAR 文件的名称具有 `.jar` 扩展名。
- J2EE 应用程序客户机 JAR：该 JAR 文件包含通过 RMI/IIOP 访问服务器端组件（如企业 Bean）的 J2EE 应用程序客户机的代码。在管理控制台中， J2EE 应用程序客户机被称为 “应用程序客户机”。 J2EE 应用程序客户机 JAR 文件的名称具有 `.jar` 扩展名。
- 资源适配器归档文件 (RAR)：RAR 文件保存资源适配器。资源适配器是按照 J2EE 连接器体系结构规范定义的，它是允许企业 Bean 和 Web 组件和应用程序客户机访问资源和外部企业系统的可移植组件。资源适配器经常称为连接器。 RAR 文件名具有 `.rar` 扩展名。

- 企业应用程序归档文件 (EAR)：EAR 文件包含一个或多个 WAR 文件、EJB JAR 文件、 RAR 文件或 J2EE 应用程序客户机 JAR 文件。 EAR 文件名具有 `.ear` 扩展名。

软件供应商可以将应用程序汇编为一个 EAR 文件或多个独立的 WAR 文件、 EJB JAR 文件和应用程序客户机 JAR 文件。在管理工具中，用于所有类型文件的部署页面和命令都是类似的。

命名约定

在给定域中，已部署的应用程序名称和模块名称必须唯一。

- 如果使用管理控制台进行部署，请在 “应用程序名称”字段中指定名称。
- 如果使用 `asadmin deploy` 命令进行部署，则应用程序或模块的默认名称为要部署的 JAR 文件的前缀。例如，如果部署 `hello.war` 文件，则 Web 应用程序的名称为 `hello`。要覆盖默认名称，请指定 `--name` 选项。

在一个应用程序中，不同类型的模块可以具有相同的名称。部署应用程序时，将使用 `_jar`、 `_war` 和 `_rar` 后缀来命名保存各个模块的目录。相同类型的模块在一个应用程序内必须具有唯一的名称。此外，数据库架构文件名在一个应用程序内必须是唯一的。

建议将 Java 类似软件包的命名模式用于模块文件名、EAR 文件名、在 `ejb-jar.xml` 文件的 `<module-name>` 部分找到的模块名以及在 `ejb-jar.xml` 文件的 `<ejb-name>` 部分找到的 EJB 名称。使用这种类似软件包的命名模式可以确保不会发生名称冲突。这种命名方式的好处不仅适用于 Sun Java System Application Server，也适用于其他 J2EE 应用程序服务器。

EJB 的 JNDI 查找名称也必须是唯一的。建立一致的命名约定可能会有帮助。例如，将应用程序名和模块名附加到 EJB 名称中是一种确保名称唯一的方式。在这种情况下， mycompany.pkging.pkgingEJB.MyEJB 将是模块 pkgingEJB.jar （该模块封装在应用程序 pkging.ear 中）中的 EJB 的 JNDI 名称。

请确保软件包和文件名称中不包含空格或操作系统视为非法的字符。

用于部署应用程序的管理控制台任务

部署企业应用程序

企业应用程序封装在 EAR 文件中，此文件是一种包含任何类型的 J2EE 独立模块（如 WAR 文件和 EJB JAR 文件）的归档文件。

要部署 （安装）企业应用程序，请执行以下步骤：

1. 在树组件中，展开 “应用程序”节点。
2. 选择 “企业应用程序”节点。
3. 在 “企业应用程序”页面中，单击 “部署”。
4. 在 “部署”页面中，指定要部署的 EAR 文件的位置。

   服务器是运行应用程序服务器和域管理服务器的主机。客户机是可以用于通过浏览器来查看管理控制台的主机。

   a. 如果文件位于客户机上或可以从客户机访问，则单击单选按钮以指定软件包文件以上载到 Application Server。

      单击 “浏览”以找到文件，或键入文件的完整路径。

   b. 如果文件位于服务器上，或者要从展开的目录部署未封装的应用程序，则单击单选按钮以指定必须可以从服务器上访问的软件包文件或目录路径。

      键入文件或目录的完整路径名。从展开的目录部署适用于高级开发者，但不建议在生产环境下进行此操作。

5. 单击 “下一步”将显示 “部署企业应用程序”页面。
6. 在 “部署企业应用程序”页面中，请指定应用程序的设置。
   a. 在 “应用程序名称”字段中，既可以保留默认名称 （即文件名的前缀），也可以键入其他名称。（如果您选择上载文件，将显示默认名称。）应用程序名称必须唯一。
   b. 默认情况下，部署应用程序之后即可使用该应用程序。要在部署之后禁用应用程序以使其不可用，请选中 “已禁用”单选按钮。
   c. 如果已部署应用程序，请选中 “重新部署”复选框以对其进行重新部署；否则将显示错误。您也可以选择其他应用程序名称，并以新名称对其进行部署。
   d. 要在部署之前检验文件的结构和内容，请选中 “验证器”复选框。大型应用程序的检验会很耗费时间。如果怀疑文件已损坏或不可移植，请检验文件。
   e. 要预编译 JSP 页面，请选中 "JSP" 复选框。如果未选中此复选框，则首次访问 JSP 页面时会在运行时编译这些页面。由于编译通常很耗费时间，因此在生产环境中请选中此复选框。
   f. 选择高可用性设置。

      要启用应用程序的高可用性，请选中 “可用性”复选框。如果启用了应用程序的可用性，则必须也在所有更高级别 （指配置和 Web 容器或 EJB 容器）启用可用性。
   g. 选择要将应用程序部署到的目标。

      从可用目标的列表中选择目标，并单击 “添加”。目标可以是群集或独立服务器实例。如果不选择目标，应用程序将被部署到默认服务器实例 `server`。

      如果您要重新部署，请勿选择目标。此时您所作的任何选择都将被忽略。引用已部署的应用程序的所有目标群集或独立服务器实例都将自动引用新的、重新部署的应用程序 （如果已启用群集或独立实例的动态重新配置）。有关如何在不中断服务的情况下重新部署应用程序的更多信息，请参见第 83 页上的 “升级应用程序”。
   h. 选择是否生成 RMI 存根。

      如果选择生成 RMI 存根，将生成静态 RMI-IIOP 存根并将其添加到 `client.jar`。
7. 单击 “确定”以部署应用程序。

等效的 `asadmin` 命令为：`deploy`

部署 Web 应用程序

Web 应用程序封装在 WAR 文件中，此文件是一种包含组件 （如 Servlet 和 JSP 页面）的归档文件。

要部署 （安装） Web 应用程序，请执行以下步骤：

1. 在树组件中，展开 “应用程序”节点。
2. 选择 “Web 应用程序”节点。
3. 在 “Web 应用程序”页面中，单击 “部署”。
4. 在 “部署”页面中，指定要部署的 WAR 文件的位置。

   服务器是运行应用程序服务器和域管理服务器的主机。客户机是可以用于通过浏览器来查看管理控制台的主机。

   a. 如果文件位于客户机上或可以从客户机访问，则单击单选按钮以指定软件包文件以上载到 Application Server。

      单击 “浏览”以找到文件，或键入文件的完整路径。

   b. 如果文件位于服务器上，或者要从展开的目录部署未封装的应用程序，则单击单选按钮以指定必须可以从服务器上访问的软件包文件或目录路径。

      键入文件或目录的完整路径名。从展开的目录部署适用于高级开发者，但不建议在生产环境下进行此操作。

5. 单击 “下一步”以显示 “部署 Web 应用程序”页面。
6. 在 “部署 Web 应用程序”页面中，指定应用程序的设置。

   a. 在 “应用程序名称”字段中，既可以保留默认名称 （即文件名的前缀），也可以键入其他名称。（如果您选择上载文件，将显示默认名称。）应用程序名称必须唯一。

   b. 在 “上下文根路径”字段中，输入标识 Web 应用程序的字符串。在 Web 应用程序的 URL 中，上下文根路径要紧跟着端口号 (`http://host:port/context-root/...`)。确保上下文根路径以正斜杠开头，例如：`/hello`

   c. 默认情况下，部署应用程序之后即可使用该应用程序。要在部署之后禁用应用程序以使其不可用，请选中 “已禁用”单选按钮。

   d. 如果已部署应用程序，请选中 “重新部署”复选框以对其进行重新部署；否则将显示错误。您也可以选择其他应用程序名称，并以新名称对其进行部署。

e. 要在部署之前检验文件的结构和内容，请选中 “验证器”复选框。大型应用程序的检验通常会很耗费时间。如果怀疑文件已损坏或不可移植，请检验文件。

f. 要预编译 JSP 页面，请选中 "JSP" 复选框。如果未选中此复选框，则首次访问 JSP 页面时会在运行时编译这些页面。由于编译通常很耗费时间，因此在生产环境中请选中此复选框。

g. 选择高可用性设置。

   要启用应用程序的高可用性，请选中 “可用性”复选框。如果启用了应用程序的可用性，则必须也在所有更高级别 （指配置和 Web 容器或 EJB 容器）启用可用性。

h. 选择要将应用程序部署到的目标。

   从可用目标的列表中选择目标，并单击 “添加”。目标可以是群集或独立服务器实例。如果不选择目标，应用程序将被部署到默认服务器实例 `server`。

   如果您要重新部署，请勿选择目标。此时您所作的任何选择都将被忽略。引用已部署的应用程序的所有目标群集或独立服务器实例都将自动引用新的、重新部署的应用程序 （如果已启用群集或独立实例的动态重新配置）。有关如何在不中断服务的情况下重新部署应用程序的更多信息，请参见第 83 页上的 “关于滚动升级”。

i. 选择是否生成 RMI 存根。

   如果选择生成 RMI 存根，将生成静态 RMI-IIOP 存根并将其添加到 `client.jar`。

7. 单击 “确定”以部署应用程序。

等效的 `asadmin` 命令为：`deploy`

启动已部署的 Web 应用程序

部署完应用程序之后，您可以从管理控制台启动该应用程序。

1. 在树组件中，展开 “应用程序”节点。
2. 单击 “Web 应用程序”。
3. 单击 Web 应用程序的 “启动”链接。
4. 在 “Web 应用程序链接”页面中单击链接以启动应用程序。

   服务器和 HTTP 侦听器必须正在运行，应用程序才能启动。

部署 EJB 模块

EJB 模块也称为 EJB JAR 文件，它包含企业 Bean。

要部署 （安装） EJB 模块，请执行以下步骤：

1. 在树组件中，展开 “应用程序”节点。
2. 选择 “EJB 模块”节点。
3. 在 “EJB 模块”页面中，单击 “部署”。
4. 在 “部署”页面中，指定要部署的 JAR 文件的位置。

   服务器是运行应用程序服务器和域管理服务器的主机。客户机是可以用于通过浏览器来查看管理控制台的主机。

   a. 如果文件位于客户机上或可以从客户机访问，则单击单选按钮以指定软件包文件以上载到 Application Server。

      单击 “浏览”以找到文件，或键入文件的完整路径。

   b. 如果文件位于服务器上，或者要从展开的目录部署未封装的应用程序，则单击单选按钮以指定必须可以从服务器上访问的软件包文件或目录路径。

      键入文件或目录的完整路径名。从展开的目录部署适用于高级开发者，但不建议在生产环境下进行此操作。

5. 单击 “下一步”以显示 “部署 EJB 模块”页面。
6. 在 “部署 EJB 模块”页面中，指定模块的设置。

   a. 在 “应用程序名称”字段中，既可以保留默认名称 （即文件名的前缀），也可以键入其他名称。（如果您选择上载文件，将显示默认名称。）应用程序名称必须唯一。

   b. 默认情况下，部署模块之后即可使用该模块。要在部署之后禁用模块以使其不可用，请选中 “已禁用”单选按钮。

   c. 如果已部署模块，请选中 “重新部署”复选框以对其进行重新部署；否则将显示错误。您也可以选择其他应用程序名称，并以新名称对其进行部署。

   d. 要在部署之前检验文件的结构和内容，请选中 “验证器”复选框。大型应用程序的检验会很耗费时间。如果怀疑文件已损坏或不可移植，请检验文件。

e. 选择高可用性设置。

要启用模块的高可用性，请选中 “可用性”复选框。如果启用了模块的可用性，则必须也在所有更高级别 （指配置和 Web 容器或 EJB 容器）启用可用性。

f. 选择要将模块部署到的目标。

从可用目标的列表中选择目标，并单击 “添加”。目标可以是群集或独立服务器实例。如果不选择目标，模块将被部署到默认服务器实例 `server`。

如果您要重新部署，请勿选择目标。此时您所作的任何选择都将被忽略。引用已部署的模块的所有目标群集或独立服务器实例都将自动引用新的、重新部署的模块 （如果已启用群集或独立实例的动态重新配置）。有关如何在不中断服务的情况下重新部署模块的更多信息，请参见第 83 页上的 “关于滚动升级”。

g. 选择是否生成 RMI 存根。

如果选择生成 RMI 存根，将生成静态 RMI-IIOP 存根并将其添加到 `client.jar`。

7. 单击 “确定”以部署该模块。

等效的 `asadmin` 命令为：`deploy`

部署连接器模块

连接器也称为资源适配器，它封装在一种称为 RAR 文件的归档文件中。

要部署 （安装）连接器模块，请执行以下步骤：

1. 在树组件中，展开 “应用程序”节点。
2. 选择 “连接器模块”节点。
3. 在 “连接器模块”页面中，单击 “部署”。
4. 在 “部署”页面中，指定要部署的 RAR 文件的位置。

服务器是运行应用程序服务器和域管理服务器的主机。客户机是可以用于通过浏览器来查看管理控制台的主机。

a. 如果文件位于客户机上或可以从客户机访问，则单击单选按钮以指定软件包文件以上载到 Application Server。

   单击 “浏览”以找到文件，或键入文件的完整路径。

b. 如果文件位于服务器上，或者要从展开的目录部署未封装的模块，则单击单选按钮以指定必须可以从服务器上访问的软件包文件或目录路径。

   键入文件或目录的完整路径名。从展开的目录部署适用于高级开发者，但不建议在生产环境下进行此操作。

5. 单击 “下一步”以显示 “部署连接器模块”页面。

6. 在 “部署连接器模块”页面中，指定模块的设置。

   a. 在 “应用程序名称”字段中，既可以保留默认名称 （即文件名的前缀），也可以键入其他名称。（如果您选择上载文件，将显示默认名称。）应用程序名称必须唯一。

   b. 在 “线程池 ID”字段中，为要部署的资源适配器指定线程池。

      默认情况下，Sun Java System Application Server 处理其默认线程池中所有资源适配器的工作请求。使用该字段可以关联特定用户创建的线程池以处理资源适配器的工作请求。

   c. 默认情况下，部署模块之后即可使用该模块。要在部署之后禁用模块以使其不可用，请选中 “已禁用”单选按钮。

      启用或禁用连接器模块时，您也同时启用或禁用了指向该模块的连接器资源和连接池。

   d. 如果已部署模块，请选中 “重新部署”复选框以对其进行重新部署；否则将显示错误。您也可以选择其他应用程序名称，并以新名称对其进行部署。

   e. 要在部署之前检验文件的结构和内容，请选中 “验证器”复选框。大型应用程序的检验通常会很耗费时间。如果怀疑文件已损坏或不可移植，请检验文件。

   f. 如果资源适配器被指定了其他属性，将显示这些属性。

      使用此表可以修改这些属性的默认值。

g. 选择要将模块部署到的目标。

从可用目标的列表中选择目标，并单击 “添加”。目标可以是群集或独立服务器实例。如果不选择目标，模块将被部署到默认服务器实例 `server`。

如果您要重新部署，请勿选择目标。此时您所作的任何选择都将被忽略。引用已部署的模块的所有目标群集或独立服务器实例都将自动引用新的、重新部署的模块 （如果已启用群集或独立实例的动态重新配置）。有关如何在不中断服务的情况下重新部署模块的更多信息，请参见第 83 页上的 “关于滚动升级”。

7. 单击 “确定”以部署该模块。

等效的 `asadmin` 命令为：`deploy`

创建生命周期模块

在被服务器生命周期中的一个或多个事件触发之后，生命周期模块将执行任务。这些服务器事件包含：

- 初始化
- 启动
- 为服务请求作准备
- 关闭

生命周期模块不符合 J2EE 规范，但是该模块是 Sun Java System Application Server 的增强功能。

要创建生命周期模块，请执行以下步骤：

1. 在树组件中，展开 “应用程序”节点。
2. 选择 “生命周期模块”节点。
3. 在 “生命周期模块”页面中，单击 “新建”。
4. 在 “创建生命周期模块”页面中，指定以下设置：
   a. 在 “名称”字段中，键入表示模块功能的名称。
   b. 在 “类名”字段中，键入生命周期模块的类文件的全限定名称。

c. 如果包含生命周期的 JAR 文件位于服务器的类路径中，则请将 “类路径” 字段保留为空。否则，键入全限定路径。

   如果不指定类路径，则必须在 *application_server_home*/domains/*domain*/applications/lifecycle-module/*module_name* 中解压缩类。如果指定类路径，则无需其他操作。

d. 在 “装入顺序” 字段中，键入大于 100 且小于操作系统的 MAXINT 值的整数。

   该整数将确定服务器启动时装入生命周期模块的顺序。系统将较早装入具有较小整数的模块。

e. 启动服务器时，服务器将装入已部署的生命周期模块。默认情况下，如果装入失败，服务器仍将继续进行启动操作。要在装入失败时禁止服务器启动，请选中 “装入失败时” 复选框。

f. 默认情况下，部署模块之后即可使用该模块。要在部署之后禁用模块以使其不可用，请选中 “已禁用” 单选按钮。

g. 选择要将模块部署到的目标。

   从可用目标的列表中选择目标，并单击 “添加”。目标可以是群集或独立服务器实例。如果不选择目标，模块将被部署到默认服务器实例 server。

   如果您要重新部署，请勿选择目标。此时您所作的任何选择都将被忽略。引用已部署的模块的所有目标群集或独立服务器实例都将自动引用新的、重新部署的模块 （如果已启用群集或独立实例的动态重新配置）。有关如何在不中断服务的情况下重新部署模块的更多信息，请参见第 83 页上的 “关于滚动升级”。

5. 单击 “确定”。

等效的 asadmin 命令为：create-lifecycle-module

部署应用程序客户机模块

应用程序客户机模块也称为 J2EE 应用程序客户机 JAR 文件，它包含客户机的服务器端例程。

要部署 （安装）应用程序客户机模块，请执行以下步骤：

1. 在树组件中，展开 “应用程序” 节点。
2. 选择 “应用程序客户机模块” 节点。
3. 在 “应用程序客户机模块” 页面中，单击 “部署”。

4. 在 “部署”页面中，指定要部署的 JAR 文件的位置。

   服务器是运行应用程序服务器和域管理服务器的主机。客户机是可以用于通过浏览器来查看管理控制台的主机。

   a. 如果文件位于客户机上或可以从客户机访问，则单击单选按钮以指定软件包文件以上载到 Application Server。

      单击 “浏览”以找到文件，或键入文件的完整路径。

   b. 如果文件位于服务器上，或者要从展开的目录部署未封装的模块，则单击单选按钮以指定必须可以从服务器上访问的软件包文件或目录路径。

      键入文件或目录的完整路径名。从展开的目录部署适用于高级开发者，但不建议在生产环境下进行此操作。

5. 单击 “下一步”以显示 “部署应用程序客户机模块”页面。

6. 在 “部署应用程序客户机模块”页面中，指定模块的设置。

   a. 在 “应用程序名称”字段中，既可以保留默认名称 （即文件名的前缀），也可以键入其他名称。（如果您选择上载文件，将显示默认名称。）应用程序名称必须唯一。

   b. 如果已部署模块，请选中 “重新部署”复选框以对其进行重新部署；否则将显示错误。您也可以选择其他应用程序名称，并以新名称对其进行部署。

   c. 要在部署之前检验文件的结构和内容，请选中 “验证器”复选框。大型应用程序的检验会很耗费时间。如果怀疑文件已损坏或不可移植，请检验文件。

   d. 选择要将模块部署到的目标。

      从可用目标的列表中选择目标，并单击 “添加”。目标可以是群集或独立服务器实例。如果不选择目标，模块将被部署到默认服务器实例 `server`。

      如果您要重新部署，请勿选择目标。此时您所作的任何选择都将被忽略。引用已部署的模块的所有目标群集或独立服务器实例都将自动引用新的、重新部署的模块 （如果已启用群集或独立实例的动态重新配置）。有关如何在不中断服务的情况下重新部署模块的更多信息，请参见第 83 页上的 “关于滚动升级”。

   e. 选择是否生成 RMI 存根。

      如果选择生成 RMI 存根，将生成静态 RMI-IIOP 存根并将其添加到 `client.jar`。

7. 单击 “确定”以部署该模块。

对于客户端例行程序：

- 通常，应用程序供应商会发行包含客户端例行程序的 JAR 文件。
- 应用程序供应商通过指定 `asadmin deploy` 命令的 `--retrieve` 选项来获取客户端存根。

等效的 `asadmin` 命令为：`deploy`

用于列出、取消部署以及启用应用程序的管理控制台任务

列出已部署的应用程序

要列出已部署的应用程序，请执行以下步骤：

1. 在树组件中，展开 “应用程序”节点。
2. 展开应用程序或模块类型的节点。

要查看已部署的应用程序或模块的详细信息，请执行以下操作之一：

- 在树组件中，选择应用程序或模块的节点。
- 在页面中选择 “应用程序名称”一栏中的条目。

等效的 `asadmin` 命令为：`list-components`

列出子组件

企业应用程序、Web 应用程序、EJB 模块和连接器模块都包含子组件。例如，Web 应用程序可能包含一个或多个 Servlet。

要列出应用程序或模块的子组件，请执行以下步骤：

1. 在树组件中，展开 “应用程序”节点。
2. 展开要查看其描述符的应用程序或模块类型的节点。
3. 选择已部署的应用程序或模块的节点。
4. 在 “应用程序或模块”页面中，查看 “子组件”表的内容。

等效的 asadmin 命令为：`list-components`

查看已部署的应用程序的模块描述符

对于企业应用程序、 Web 应用程序、 EJB 模块、连接器模块和应用程序客户机模块，您可以查看模块部署描述符。

1. 在树组件中，展开 “应用程序”节点。
2. 选择要查看其描述符的应用程序或模块类型的节点。
3. 选择已部署的应用程序或模块的节点。
4. 选择 “描述符”选项卡。
5. 要查看描述符文件的文本，请单击文件名。

   页面将显示文件内容。此信息为只读。

取消部署应用程序

取消部署应用程序或模块将从域中将其卸载并删除所有实例对它的引用。

要取消部署应用程序或模块，请执行以下步骤：

1. 在树组件中，展开 “应用程序”节点。
2. 选择要取消部署的应用程序或模块的类型的节点。

3. 在列出已部署的应用程序的表中，选中要取消部署的应用程序或模块的复选框。
4. 单击 “取消部署”。

等效的 asadmin 命令为：undeploy

启用和禁用应用程序

如果启用了已部署的应用程序或模块，则可以通过客户机对其进行访问。如果将其禁用，它将仍然处于部署状态，但不能通过客户机对其进行访问。默认情况下，在部署应用程序或模块时，由于默认情况下选中了 “在所有目标上启用”单选按钮，因此将启用该应用程序或模块。

要启用已部署的应用程序或模块，请执行以下步骤：

1. 在树组件中，展开 “应用程序”节点。
2. 展开应用程序类型的节点。
3. 选中已部署的应用程序或模块旁边的复选框。
4. 单击 “启用”或 “禁用”。

   这些按钮用于在所有目标上启用或禁用应用程序。

要在单个目标上启用应用程序，请执行以下步骤：

1. 在树组件中，展开 “应用程序”节点。
2. 展开应用程序类型的节点。
3. 选择应用程序的节点。
4. 单击 “目标”选项卡。
5. 选中已部署的应用程序或模块旁边的复选框。
6. 单击 “启用”或 “禁用”。

等效的 asadmin 命令为：enable 和 disable

管理应用程序目标

部署应用程序或模块后，可通过管理目标来管理引用该应用程序或模块的服务器实例和群集。

1. 在树组件中，展开 “应用程序”节点。
2. 展开应用程序类型的节点。
3. 选择已部署的应用程序的节点。
4. 选择 “目标”选项卡。
5. 要在特定目标实例或群集上启用或禁用应用程序，请单击目标旁边的复选框并单击 “启用”或 “禁用”。
6. 要添加或删除应用程序的目标，则选择 “管理目标”。
7. 添加或删除目标并单击 “确定”。

   此时，已修改的目标列表中将包含该应用程序。

等效的 asadmin 命令为：create-application-ref 和 delete-application-ref。

部署在其他虚拟服务器上

将应用程序或模块部署到目标服务器实例或群集之后，可以将其与其他虚拟服务器关联。

1. 在已部署的应用程序或模块的 “目标”页面中，单击目标旁边的 “管理虚拟服务器”链接。
2. 在可用虚拟服务器的列表中，添加或删除虚拟服务器目标。
3. 单击 “确定”。

重新部署到多个目标

如果将应用程序部署到多个目标 （独立服务器实例或群集），则可以通过两种方式来重新部署到多个目标。使用以下方法之一可以确保引用应用程序的所有服务器实例接收到最新版本。

开发环境

在开发环境中，只重新部署应用程序。应用程序将被重新部署到域，并且引用它的所有目标将自动接收到新的版本 （如果已启用目标服务器实例的动态重新配置）。默认情况下，将启用动态重新配置。如果未启用服务器实例的动态重新配置，将继续使用旧版本，直至重新启动服务器实例。

生产环境

在生产环境中，请按第 83 页上的 “关于滚动升级”中所述的步骤进行操作。

启用和禁用动态重新装入

如果启用了动态重新装入，服务器将定期检查已部署的应用程序中的更改并自动重新装入包含更改的应用程序。更改将通过您手动创建的名为 `.reload` 的文件的日期更改来显示。应用程序必须安装在 *server_root*`/domain/domain1/applications/`*j2ee-module_or_j2ee-apps*`/`*app_or_module_name* 中。

例如：`AppServer/domain/domain1/applications/j2ee-module/webapps-simple`

动态重新装入在开发环境中很有用，因为它允许快速测试代码更改。但在生产环境中，动态重新装入可能会使性能降低。

注 动态重新装入仅适用于默认服务器实例。

动态重新装入适用于开发环境。它与会话持久性 （一种生产环境功能）不兼容。如果启用了动态部署，请勿启用会话持久性。

要配置动态重新装入，请执行以下步骤：

1. 在树组件中，展开 “独立实例”节点。
2. 单击 "server" （管理服务器）。
3. 单击 “高级”。
4. 在 “应用程序配置”页面中，配置以下项目：
   - 重新装入：使用 “已启用”复选框来启用或禁用动态重新装入。
   - 重新装入轮询时间间隔：指定服务器检查已部署的应用程序中的更改的频率。
   - 管理会话超时：指定管理会话超时且必须重新登录之前的时间。

将系统配置为使用动态重新装入之后，请为每个要动态重新装入的应用程序创建一个名为 .reload 的文件并将其置于应用程序的目录中。此文件没有任何内容。更改应用程序时，将更改此文件的日期 （例如，使用 UNIX `touch` 命令），并且将自动重新装入更改。

适用于开发者的部署方法

使用自动部署

自动部署功能使您能够将预封装的应用程序或模块复制到 *domain_root_dir*/*domain_dir*/`autodeploy` 目录来部署该应用程序或模块。

例如，将名为 `hello.war` 的文件复制到 *domain_root_dir*/`domain1/autodeploy` 目录。要取消部署应用程序，请从 `autodeploy` 目录中删除 `hello.war` 文件。

您也可以使用管理控制台或 asadmin 工具来取消部署应用程序。在这种情况下，归档文件将保留。

自动部署功能适用于开发环境。它与会话持久性 （一种生产环境功能）不兼容。如果启用了动态部署，请勿启用会话持久性。

注 自动部署仅适用于默认服务器实例。

要配置自动部署功能，请执行以下步骤：

1. 在树组件中，展开 “独立实例”节点。
2. 单击 "server" （管理服务器）。
3. 单击 “高级”。
4. 在 “应用程序配置”页面中，配置以下项目：
   a. 通过选中或取消选中 “已启用”复选框来启用或禁用自动部署。
   b. 在 “自动部署轮询时间间隔”字段中，指定服务器检查自动部署目录中的应用程序文件或模块文件的频率。更改轮询时间间隔不会影响部署应用程序或模块所需的时间。
   c. 在 “自动部署目录”中，如果指定建立应用程序的目录，则不必将文件复制到默认自动部署目录中。

      默认目录是服务器实例的根目录中名为 `autodeploy` 的目录。默认情况下，可使用变量自动更改多个服务器实例的目录。有关这些变量的更多信息，请参见 “设置域属性”。
   d. 要在部署之前运行验证器，请选中 “已启用验证器”复选框。验证器将检查文件的结构和内容。大型应用程序的检验通常会很耗费时间。
   e. 要预编译 JSP 页面，请选中 "JSP" 复选框。如果未选中此复选框，则首次访问 JSP 页面时会在运行时编译这些页面。由于编译通常很耗费时间，因此在生产环境中请选中此复选框。

部署目录中未封装的应用程序

此功能适用于高级开发者。

使用目录部署仅部署到默认服务器实例 (server)。您不能使用它来部署到群集或独立服务器实例。

包含未封装的应用程序或模块的目录有时称为展开的目录。目录的内容必须与相应的 J2EE 归档文件的内容匹配。例如，如果部署某一目录中的 Web 应用程序，则该目录的内容必须与相应的 WAR 文件的内容相同。有关必需的目录内容的信息，请参见相应的规范。

您可以直接在展开的目录中更改部署描述符文件。

如果您的环境被配置为使用动态重新装入，则还可以从目录中动态重新装入已部署的应用程序。有关更多信息，请参见第 123 页上的 “启用和禁用动态重新装入”。

要部署目录中未封装的应用程序，请执行以下步骤：

1. 在管理控制台中启动部署进程。请参见第 111 页上的 “部署 Web 应用程序”。
2. 在 “部署”页面中，指定以下内容：
   a. 单击单选按钮以指定必须可以从服务器上访问的软件包文件或目录路径。
   b. 在 “文件或目录”字段中，输入展开的目录的名称。

等效的 asadmin 命令为：deploydir

使用 deploytool 实用程序

为软件开发者设计的 deploytool 实用程序可以封装和部署 J2EE 应用程序和模块。有关如何使用 deploytool 的说明，请参见 《The J2EE 1.4 Tutorial》。

使用部署规划

此功能适用于高级开发者。

部署规划是指仅包含特定于 Application Server 的部署描述符的 JAR 文件。有关这些部署描述符 （例如 sun-application.xml）的说明，请参见 Application Server Developer's Guide。部署规划为 *JSR 88:* J2EE Application Deployment 的实现的一部分。使用部署规划可以部署不包含特定于 Application Server 的部署描述符的应用程序或模块。

要使用部署规划进行部署，请指定 asadmin deploy 命令中的 --deploymentplan 选项。例如，以下命令将根据 mydeployplan.jar 文件中指定的规划来部署 myrosterapp.ear 文件中的企业应用程序。

```
$ asadmin deploy --user admin ---deploymentplan mydeployplan.jar
myrosterapp.ear
```

在企业应用程序 (EAR) 的部署规划文件中， sun-application.xml 文件位于根目录下。根据以下语法来存储每个模块的部署描述符：*module-name.sun-dd-name*，其中的 *sun-dd-name* 取决于模块类型。如果模块包括 CMP 映射文件，则该文件命名为 *module-name*.sun-cmp-mappings.xml。 .dbschema 文件存储在根级别目录下，并用井号 (#) 替换每个正斜杠符号 (/)。下面列出的内容显示了企业应用程序 (EAR) 的部署规划文件的结构。

```
$ jar -tvf mydeployplan.jar
420 Thu Mar 13 15:37:48 PST 2003 sun-application.xml
370 Thu Mar 13 15:37:48 PST 2003 RosterClient.war.sun-web.xml
418 Thu Mar 13 15:37:48 PST 2003 roster-ac.jar.sun-application-client.xml
1281 Thu Mar 13 15:37:48 PST 2003 roster-ejb.jar.sun-ejb-jar.xml
2317 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-ejb-jar.xml
3432 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-cmp-mappings.xml
84805 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.RosterSchema.dbschema
```

在 Web 应用程序或模块文件的部署规划中，特定于 Application Server 的部署描述符位于根级别目录下。如果独立 EJB 模块包括 CMP Bean，则部署规划包括位于根级别目录的 `sun-cmp-mappings.xml` 和 `.dbschema` 文件。在下面列出的内容中，部署规划描述了 CMP Bean。

```
$ jar r -tvf myotherplan.jar
3603 Thu Mar 13 15:24:20 PST 2003 sun-ejb-jar.xml
3432 Thu Mar 13 15:24:20 PST 2003 sun-cmp-mappings.xml
84805 Thu Mar 13 15:24:20 PST 2003 RosterSchema.dbschema
```

第 6 章

JDBC 资源

本章说明如何配置访问数据库的应用程序所需的 JDBC 资源。本章包含以下几节：

- 关于 JDBC 资源
- 设置数据库访问
- 关于 JDBC 连接池
- 关于 JDBC 资源
- 关于 Persistence Manager 资源

关于 JDBC 资源

- JDBC 资源
- JDBC 连接池
- JDBC 资源和连接池如何协同工作

JDBC 资源

为了存储、组织和检索数据，大多数应用程序都采用了关系数据库。 J2EE 应用程序通过 JDBC API 访问关系数据库。

JDBC 资源 （数据源）为应用程序提供了连接数据库的方法。通常，管理员要为部署在域中的应用程序访问的每个数据库创建 JDBC 资源。（但是，可以为一个数据库创建多个 JDBC 资源。）

要创建 JDBC 资源，请指定标识资源的唯一 JNDI 名称。（请参见 “JNDI 名称和资源”部分。） JDBC 资源的 JNDI 名称应在 `java:comp/env/jdbc` 子上下文中。例如，工资单数据库资源的 JNDI 名称可以为 `java:comp/env/jdbc/payrolldb`。由于所有资源 JNDI 名称位于 `java:comp/env` 子上下文中，所以在管理控制台中指定 JDBC 资源的 JNDI 名称时仅需输入 `jdbc/`*`name`*。例如，对于工资单数据库，可以指定 `jdbc/payrolldb`。

JDBC 连接池

要创建 JDBC 资源，请指定与其关联的连接池。多个 JDBC 资源可以指定一个连接池。

JDBC 连接池是用于特定数据库的一组可重复使用的连接。由于每创建一个新的物理连接都会耗费时间，因此服务器维护了可用连接池以提高性能。应用程序请求连接时可以从池中获取一个连接。应用程序关闭连接时，连接将返回到池中。

连接池的属性可能会随数据库供应商的不同而有所不同。有一些属性是通用的，如数据库名称 (URL)、用户名和密码。

JDBC 资源和连接池如何协同工作

为了存储、组织和检索数据，大多数应用程序都采用了关系数据库。 J2EE 应用程序通过 JDBC API 访问关系数据库。应用程序必须获得一个连接之后才可以访问数据库。

以下是运行时应用程序连接到数据库时所发生的情况：

1. 应用程序通过 JNDI API 进行调用获取与数据库关联的 JDBC 资源 （数据源）。

   给定资源的 JNDI 名称、命名和目录服务定位 JDBC 资源。每个 JDBC 资源指定一个连接池。

2. 通过 JDBC 资源，应用程序获得一个数据库连接。

   应用程序服务器秘密地从与该数据库相对应的连接池中检索物理连接。池定义了数据库名称 (URL)、用户名和密码等连接属性。

3. 由于已将应用程序连接到数据库，所以该应用程序可以读取和修改数据库中的数据以及将数据添加到数据库中。

   应用程序通过对 JDBC API 进行调用来访问数据库。 JDBC 驱动程序将应用程序的 JDBC 调用转换为数据库服务器的协议。

4. 访问数据库完成之后，应用程序将关闭该连接。

   应用程序服务器将连接返回连接池。连接返回连接池之后，下一个应用程序就可以使用该连接。

设置数据库访问

设置数据库访问的一般步骤

1. 安装支持的数据库产品。有关 Application Server 支持的数据库产品的列表，请参见 “更多信息”部分中指向发行说明的链接。
2. 安装适用于该数据库产品的 JDBC 驱动程序。
3. 使域的服务器实例可以访问此驱动程序的 JAR 文件。请参见集成 JDBC 驱动程序。
4. 创建数据库。通常，应用程序供应商提供了用于创建和填充数据库的脚本。
5. 为数据库创建连接池。请参见创建 JDBC 连接池。
6. 创建指向连接池的 JDBC 资源。请参见创建 JDBC 资源。

集成 JDBC 驱动程序

JDBC 驱动程序将应用程序的 JDBC 调用转换为数据库服务器的协议。要将 JDBC 驱动程序集成到管理域中，请执行以下操作之一：

- 使通用类加载器可以访问该驱动程序。
  - 将驱动程序的 JAR 文件和 ZIP 文件复制到 *install_dir*`/domains/`*domain_dir*`/lib` 目录或将驱动程序的类文件复制到 *install_dir*`/domains/`*domain_dir*`/lib/ext` 目录中。
  - 重新启动该域。

- 使系统类加载器可以访问驱动程序。
  - 在管理控制台的树视图中，选择 “配置”。
  - 选择所需的配置 （例如 default-config）。
  - 选择 “JVM 设置”。
  - 在 “JVM 设置”页面中，单击 “路径设置”选项卡。
  - 在 “类路径后缀”字段中，输入驱动程序 JAR 文件的全限定路径名。
  - 单击 “保存”。
  - 重新启动服务器。

关于 JDBC 连接池

创建 JDBC 连接池

JDBC 连接池是用于特定数据库的一组可重复使用的连接。使用管理控制台创建池时，管理员实际上在定义到特定数据库的连接的各个方面。

创建池之前，您必须首先安装并集成 JDBC 驱动程序。

设置 “创建连接池”页面时，必须输入特定于 JDBC 驱动程序和数据库供应商的特定数据。继续创建之前，请先收集以下信息：

- 数据库供应商名称
- 资源类型，如 `javax.sql.DataSource` （仅限于本地事务）和 `javax.sql.XADataSource` （全局事务）
- 数据源类名
- 必需的属性，如数据库名称 (URL)、用户名和密码

要创建 JDBC 连接池，请执行以下步骤：

1. 在树组件中，展开 “资源”节点。
2. 在 “资源”节点下，展开 "JDBC" 节点。
3. 在 "JDBC" 节点下，选择 “连接池”节点。
4. 在 “连接池”页面中，单击 “新建”。
5. 在 “创建连接池”的第一个页面中，指定以下常规设置：
   a. 在 “名称”字段中，输入池的逻辑名称。

      创建 JDBC 资源时将指定此名称。

   b. 从 “资源类型”组合框中选择一个条目。
   c. 从 “数据库供应商”组合框中选择一个条目。
6. 单击 “下一步”。
7. 在 “创建连接池”的第二个页面中，为 “DataSource 类名”字段指定值。

   如果 JDBC 驱动程序具有与在上一页中指定的资源类型和数据库供应商相应的 DataSource 类，则系统会提供 “DataSource 类名”字段的值。
8. 单击 “下一步”。
9. 在 “创建连接池”的第三个也即最后一个页面中，执行以下任务：
   a. 在 “常规设置”部分中，检验各个值是否正确。
   b. 对于 “池设置”、“连接验证”和 “事务隔离”部分中的字段，保留默认值。

      以后更改这些设置非常方便。请参见 “编辑 JDBC 连接池”。
   c. 在 “其他属性”表中，添加必需的属性，如数据库名称 (URL)、用户名和密码。
10. 单击 “完成”。

等效的 `asadmin` 命令为：`create-jdbc-connection-pool`

编辑 JDBC 连接池

“编辑 JDBC 连接池”页面使您可以更改现有池的所有设置 （池的名称除外）。

要访问 “编辑 JDBC 连接池”页面，请执行以下步骤：

1. 在树组件中，展开 “资源”节点。
2. 在 “资源”节点下，展开 "JDBC" 节点。
3. 在 "JDBC" 节点下，展开 “连接池”节点。
4. 选择要编辑的池的节点。
5. 在 “编辑 JDBC 连接池”页面中进行必要的更改。

   有关可以更改的设置的说明，请参见以下各个部分。
6. 单击 “保存”。

常规设置

常规设置的值取决于安装的特定 JDBC 驱动程序。这些设置是 Java 编程语言中的类名或接口名称。

表 6-1 JDBC 连接池的常规设置

参数	说明
DataSource 类名	实现 DataSource/ConnectionPoolDataSource/XADataSource API 的特定于供应商的类名。该类位于 JDBC 驱动程序中。
资源类型	选项包括 javax.sql.DataSource （仅限于本地事务）、javax.sql.XADataSource （全局事务）和 java.sql.ConnectionPoolDataSource （本地事务，性能可能会提高）。

池设置

一组物理数据库连接保存在池中。应用程序请求连接时，将从池中删除该连接；而应用程序释放该连接之后，连接将返回到池中。

表 6-2 JDBC 连接池的池设置

参数	说明
初始和最小池大小	池中连接的最小数目。该值还确定了首次创建池或应用程序服务器启动时被置于池中的连接的数目。
最大池大小	池中连接的最大数目。
池大小调整数量	当池向最小池大小方向收缩时，将成批调整大小。此值确定批处理中的连接数目。如果将该值设置得过大，会延迟连接回收；如果将该值设置得过小，则会降低效率。
空闲超时	连接在池中保持空闲的最长时间 （以秒为单位）。一旦超过此时间，即从池中删除该连接。
最长等待时间	在达到连接超时之前，请求连接的应用程序所等待的时间。由于默认等待时间过长，应用程序可能会出现无限期挂起的情况。

连接验证

（可选）应用程序服务器可以在将连接传送给应用程序之前验证连接。如果由于网络出现故障或数据库服务器崩溃造成数据库不可用，此验证将允许应用程序服务器自动重新建立数据库连接。连接验证会带来额外开销，并会导致性能稍有下降。

表 6-3 JDBC 连接池的连接验证设置

参数	说明
连接验证	选中 “需要”复选框以启用连接验证。
验证方法	应用程序服务器可以使用三种方法来验证数据库连接：自动提交、元数据和表。 • 自动提交和元数据－应用程序服务器通过调用 con.getAutoCommit() 方法和 con.getMetaData() 方法来验证连接。但是，由于许多 JDBC 驱动程序高速缓存了这些调用的结果，因此这两种方法无法始终提供可靠的验证。请与驱动程序供应商进行核实，以确定这些调用是否被高速缓存。 • 表橐应用程序将查询指定的数据库表。表必须存在并且可以访问，但不要求表的行数。请不要使用包含许多行的现有表或经常访问的表。
表的名称	如果从 “验证方法”组合框中选择了表，请在此指定数据库表的名称。
一旦失败	如果选中标有 “关闭所有连接”的复选框，则单个连接失败时，应用程序服务器将关闭池中的所有连接，然后重新建立这些连接。如果未选中此复选框，则仅当要使用各个连接时才会重新建立连接。

事务隔离

由于许多用户通常可以并行访问一个数据库，因此可能出现一个事务在更新数据而另一个事务尝试读取同一数据的情况。事务的隔离级别定义了正在更新的数据对于其他事务的可见程度。有关隔离级别的详细资料，请参见数据库供应商的文档。

表 6-4 JDBC 连接池的事务隔离设置

参数	说明
事务隔离	使您可以为该池的连接选择事务隔离级别。如果不指定此参数，连接将使用 JDBC 驱动程序提供的默认隔离级别进行操作。
保证隔离级别	该项仅在指定了隔离级别的情况下才适用。如果选中 “保证”复选框，则从池中获取的所有连接都具有相同的隔离级别。例如，如果上次使用连接时通过编程方式 （使用 `con.setTransactionIsolation`）更改了连接的隔离级别，此机制会将状态更改回指定的隔离级别。

属性

在 “其他属性”表中，可以指定数据库名称 (URL)、用户名和密码等属性。由于随数据库供应商的不同，属性也会有所不同，因此请参考供应商的文档以了解有关详细信息。

检验连接池设置

要检验连接池设置，请执行以下步骤：

1. 启动数据库服务器。
2. 单击 "Ping"。

   管理控制台将尝试连接到数据库。如果显示错误消息，请检查数据库服务器是否已重新启动。

删除 JDBC 连接池

1. 在树组件中，展开 “资源”节点。
2. 在 “资源”节点下，展开 "JDBC" 节点。
3. 在 "JDBC" 节点下，选择 “连接池”节点。
4. 在 “连接池”页面中，选中要删除的池的复选框。
5. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-jdbc-connection-pool`

关于 JDBC 资源

创建 JDBC 资源

JDBC 资源 （数据源）为应用程序提供了连接数据库的方法。创建 JDBC 资源之前，请先创建 JDBC 连接池。

要创建 JDBC 资源，请执行以下步骤：

1. 在树组件中，展开 “资源”节点。
2. 在 “资源”节点下，展开 "JDBC" 节点。
3. 在 "JDBC" 节点下，选择 “JDBC 资源”节点。
4. 在 “JDBC 资源”页面中，单击 “新建”。

5. 在 “创建 JDBC 资源”页面中，指定资源的设置：
   a. 在 “JNDI 名称”字段中，键入唯一的名称。按照约定，该名称应以 `jdbc/` 字符串开头。例如：`jdbc/payrolldb`。请不要忘记正斜杠。
   b. 从 “池名称”组合框中选择要与新的 JDBC 资源关联的连接池。
   c. 默认情况下，创建资源之后立即可以使用资源 （已启用）。如果要使资源不可用，请取消选中 “已启用”复选框。
   d. 在 “说明”字段中，键入资源的简短说明。
   e. 在 “目标”部分，指定具有可用资源的目标 （群集和独立服务器实例）。在左侧选择所需的目标，然后单击 “添加”以将其添加到选定目标的列表中。
6. 单击 “确定”。

等效的 `asadmin` 命令为：`create-jdbc-resource`

编辑 JDBC 资源

1. 在树组件中，展开 “资源”节点。
2. 在 “资源”节点下，展开 "JDBC" 节点。
3. 在 "JDBC" 节点下，展开 “JDBC 资源”节点。
4. 选择要编辑的 JDBC 资源的节点。
5. 在 “编辑 JDBC 资源”页面中，可以执行以下任务：
   a. 从 “池名称”组合框中选择其他连接池。
   b. 在 “说明”字段中，更改资源的简短说明。
   c. 选中或取消选中复选框以启用或禁用资源。

d. 选择 “目标”选项卡，以更改具有可用资源的目标 （群集和独立服务器实例）。

   选中列表中的现有目标的复选框，然后单击 “启用”以启用该目标的资源，或者单击 “禁用”以禁用该目标的资源。

   单击 “管理目标”可以在列表中添加或删除目标。在 “管理目标”页面中，从左侧的 “可用”列表中选择所需的目标，然后单击 “添加”以将其添加到选定目标的列表中。单击 “删除”可以从 “已选择”列表中删除目标。

   单击 “确定”可以保存对可用目标的更改。

6. 单击 “保存”以应用编辑。

删除 JDBC 资源

1. 在树组件中，展开 “资源”节点。
2. 在 “资源”节点下，展开 "JDBC" 节点。
3. 在 "JDBC" 节点下，选择 “JDBC 资源”节点。
4. 在 “JDBC 资源”页面中，选取要删除的资源的复选框。
5. 单击 “删除”。

- 启用和禁用 JDBC 资源

启用和禁用 JDBC 资源

1. 在树组件中，展开 “JDBC 资源”节点或展开 “独立实例”以选择 “服务器实例节点资源”选项卡。
2. 在 “资源”页面中，选中要启用或禁用的资源的复选框。
3. 单击 “启用”或 “禁用”。

关于 Persistence Manager 资源

- 创建 Persistence Manager 资源

创建持久性管理器资源

向后兼容需要此功能。要在版本 7 的 Application Server 上运行，使用容器管理的持久性 Bean （一种 EJB 组件）的应用程序需要持久性管理器资源。

要创建 Persistence Manager 资源，请执行以下步骤：

1. 在树组件中，展开 “资源”节点。
2. 在 “资源”节点下，选择 “持久性管理器”节点。
3. 在 “持久性管理器”页面中，单击 “新建”。
4. 在 “创建 Persistence Manager”页面中，指定以下设置：
   a. 在 “JNDI 名称”字段中，键入唯一的名称，例如：`jdo/mypm`。请不要忘记正斜杠。
   b. 在 “工厂类”字段中，保留此版本提供的默认类，或键入其他实现类。
   c. 从 “连接池”组合框中选择新的 Persistence Manager 资源所属的连接池。
   d. 默认情况下，将启用新的持久性管理器资源。要将其禁用，请取消选中 “已启用”复选框。
   e. 在 “目标”部分，指定具有可用资源的目标 （群集和独立服务器实例）。在左侧选择所需的目标，然后单击 “添加”以将其添加到选定目标的列表中。
5. 单击 “确定”。

等效的 `asadmin` 命令为：`create-persistence-resource`

编辑持久性管理器资源

要编辑现有的持久性管理器资源属性，请执行以下步骤：

1. 在 “编辑持久性管理器属性”选项卡中，选择 “添加属性”按钮。

   “其他属性”表中将添加一个新行。
2. 添加所需的属性和值。

管理资源目标

要管理资源目标，请执行以下步骤：

1. 选择 “目标”选项卡，以更改资源所在的目标 （群集和独立服务器实例）。
2. 选中列表中的现有目标的复选框，然后单击 “启用”以启用该目标的资源，或者单击 “禁用”以禁用该目标的资源。
3. 单击 “管理目标”可以在列表中添加或删除目标。在 “管理目标”页面中，从左侧的 “可用”列表中选择所需的目标，然后单击 “添加”以将其添加到选定目标的列表中。单击 “删除”可以从 “已选择”列表中删除目标。
4. 单击 “确定”可以保存对可用目标的更改。
5. 单击 “保存”。

删除持久性管理器资源

1. 在树组件中，展开 “持久性管理器”节点。
2. 选择 “持久性管理器”节点。
3. 在 “持久性管理器”页面中，选中要删除的持久性管理器的复选框。
4. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-persistence-resource`

启用和禁用持久性管理器资源

1. 在树组件中，展开 “持久性管理器”节点。
2. 选中要启用或禁用的资源的复选框。
3. 单击 “启用”或 “禁用”。

第 7 章

配置可用性和会话持久性

本章介绍了如何在 Sun Java™ System Application Server Enterprise Edition 环境中配置会话持久性和可用性。它包含以下小节：

- 关于可用性和会话持久性
- 有关配置可用性的管理控制台任务

关于可用性和会话持久性

- 需要会话持久性的原因
- 会话持久性配置概述
- 可用性的级别
- 处于 HTTP 会话状态的单点登录的可用性
- 样例应用程序

需要会话持久性的原因

在应用程序会话进行时，属于会话的数据常常不会存储在传统的数据库中。例如，购物车的内容就是这样的数据。 Sun Java System Application Server 提供了在系统信息库中保存或保留此会话数据的功能，这样，如果应用程序服务器实例出现故障，会话状态可以恢复并且会话可以继续进行而不会丢失信息。

在 J2EE 应用程序中，会话数据通常存储在 HTTP 会话或有状态会话 Bean (SFSB) 会话中。 Sun Java System Application Server 同时支持 HTTP 会话和 SFSB 会话的状态持久性。还支持同时存储在 HTTP 会话和 SFSB 会话中的某些 J2EE 对象参考的故障转移，请参见 Developer's Guide。

具有高可用性的数据库 (HADB) 与 Sun Java System Application Server 捆绑在一起，可用作持久性存储库以便为会话数据提供高可用性。

会话持久性配置概述

为了成功完成会话持久性配置，请确保按其所列顺序执行以下步骤，因为前面的一个或多个步骤将作为执行后面一些步骤的先决条件。

1. 为群集创建 HADB 数据库。请参见 Reference Manual 中的 `configure-ha-cluster` 命令的描述。
2. 为群集设置 HTTP 负载平衡。请参见第 3 章 “配置负载平衡和故障转移”。
3. 为应用程序服务器实例和 Web 或 EJB 容器 （应支持会话持久性）启用可用性并配置会话持久性设置。选择以下方法之一：
   - 请参见第 146 页上的 “有关配置可用性的管理控制台任务”。
   - 请参见 Reference Manual 中的 `configure-ha-persistence` 命令的描述。
4. 如果您**不**启用可用性，则可以为 SFSB 更改文件系统会话存储 （如果需要）。请参见第 146 页上的 “在禁用可用性时配置 SFSB 会话存储”。
5. 重新启动群集中的每个服务器实例。
6. 为需要可用性的任何特定 SFSB 启用可用性，然后选择需要为其进行会话状态检查点操作的方法。请参见 Developer’s Guide。
7. 使每个应具有高可用性的 Web 模块可分布。请参见 Developer’s Guide。
8. 在部署期间，为 J2EE 应用程序、Web 模块或 EJB 模块启用可用性。在管理控制台中，选取 “启用可用性” 框，或结合使用 `deploy` 命令和设置为 `true` 的 `--availabilityenabled` 选项。

注 会话持久性与动态部署、动态重新装入和自动部署不兼容。这些部署功能适用于开发环境而非生产环境。有关如何禁用这些功能的信息，请参见第 5 章 “部署应用程序”。

注 如果实例当前正在处理请求，请在重新启动该实例前停止它，这样，它就有足够的时间来处理它正在处理的请求。有关更多信息，请参见“禁用 （停止）服务器实例或群集”。

可用性的级别

可用性可在五个不同级别启用：

1. 服务器实例，默认情况下已启用
2. Web 或 EJB 容器，默认情况下已启用
3. 应用程序，默认情况下已禁用
4. 独立的 Web 或 EJB 模块，默认情况下已禁用
5. SFSB，默认情况下已禁用

要启用给定级别的可用性，它必须已在所有更高级别中启用。例如，要启用应用程序级别的可用性，必须在服务器实例级别和容器级别启用可用性。

给定级别的默认值是其上一级别的设置值。例如，如果已启用容器级别的可用性，则默认情况下，启用应用程序级别的可用性。

如果禁用了服务器实例级别的可用性，则启用其他任何级别的可用性都不生效。如果启用了服务器实例级别的可用性，则将启用所有级别的可用性，除非已明确禁用。

处于 HTTP 会话状态的单点登录的可用性

在单个应用程序服务器实例中，如果一个应用程序验证了用户，则运行在同一个实例上的其他应用程序不会对用户进行重新验证。这称为**单点登录**。有关单点登录的更多信息，请参见第 214 页上的 “检验单点登录”。

为了使 HTTP 会话故障转移到群集中另一个实例后该功能仍然可用，必须将单点登录信息保留在 HADB 中。首先启用服务器实例和 Web 容器的可用性，然后启用单点登录状态持久性。请参见第 147 页上的 “配置服务器实例级别的可用性”。

可以通过单个名称和密码的组合进行访问的应用程序组成了**单点登录组**。

对于与应用程序 （是单点登录组的一部分）相对应的 HTTP 会话，如果其中一个会话超时，其他会话并不会失效，并且仍然可用。这是因为一个会话的超时不应影响其他会话的可用性。

作为这个行为的推论，如果一个会话超时并且您尝试从运行该会话的同一浏览器窗口访问相应的应用程序，则您无需再次进行验证。但是，将创建一个新的会话。

以属于含有其他两个应用程序的单点登录组的购物车应用程序为例。假设其他两个应用程序的会话超时值大于购物车应用程序的会话超时值。如果购物车应用程序的会话超时，并且您尝试从运行该会话的同一浏览器窗口运行购物车应用程序，则您无需再次进行验证。但是，前一个购物车将丢失，并且您必须创建一个新的购物车。即使运行购物车应用程序的会话已超时，其他两个应用程序也会继续照常运行。

类似地，可假定与其他两个应用程序中的任何一个应用程序对应的会话超时。当从运行该会话的同一浏览器窗口连接应用程序时，您无需再次进行验证。

注 此行为仅适用于会话超时情况。如果启用了单点登录并且您使用 `HttpSession.invalidate()` 令其中一个会话失效，则属于单点登录组的所有应用程序的会话都将失效。如果您尝试访问属于单点登录组的任一应用程序，则需要再次进行验证，系统将为访问该应用程序的客户机创建一个新的会话。

样例应用程序

以下目录包含用于演示 HTTP 和 SFSB 会话持久性的样例应用程序：

install_dir`/samples/ee-samples/highavailability`
install_dir`/samples/ee-samples/failover`

有关配置可用性的管理控制台任务

- 在禁用可用性时配置 SFSB 会话存储
- 配置服务器实例级别的可用性
- 配置 Web 容器级别的可用性
- 配置 EJB 容器级别的可用性

在禁用可用性时配置 SFSB 会话存储

如果禁用了可用性，本地文件系统将用于 SFSB 状态钝化，但不是持久性。要更改 SFSB 状态的存储位置，请更改 EJB 容器中的 “会话存储位置” 设置。请参见第 201 页上的 “配置常规 EJB 设置”。

配置服务器实例级别的可用性

要使用管理控制台启用或禁用服务器实例级别的可用性，请执行以下步骤：

1. 在树组件中，展开 “配置”节点。
2. 展开要编辑的配置的节点。
3. 选择 “可用性服务”节点。
4. 转至 “可用性服务”页面。
5. 通过选中 “可用性服务”框来启用实例级别可用性。要禁用实例级别可用性，请取消选中该框。

   如果更改了用于连接 HADB 的 JDBC 资源以获得会话持久性，您可以更改存储池名称。有关详细信息，请参见 Reference Manual 中的 `configure-ha-cluster` 命令的描述。
6. 单击 “保存”按钮。
7. 展开 “实例”节点。
8. 选择服务器实例。
9. 转至服务器实例页面。
10. 重新启动服务器。

配置 Web 容器级别的可用性

要启用可用性或覆盖单个 Web 应用程序的可用性设置，请使用 `sun-web.xml` 文件中的设置。有关详细信息，请参见 Developer’s Guide。

要使用管理控制台启用或禁用 Web 容器的可用性，请执行以下步骤：

1. 选择 “Web 容器可用性”选项卡，然后选中 “可用性服务”框。要禁用可用性，请取消选中该框。您还可以更改以下可选设置：

   - 持久性类型：为已启用可用性的 Web 应用程序指定会话持久性机制。允许的值包括 `memory`（无持久性）、 `file`（文件系统）和 `ha` (HADB)。如果启用了可用性，则默认值为 `ha`。如果禁用了可用性，则默认值为 `memory`。对于需要会话持久性的生产环境，请使用 `ha`。

     如果将持久性类型设置为 `memory`，您可以使用 `sessionFilename` 属性来指定正常关闭服务器实例时存储 HTTP 会话状态的文件系统位置。这对于内部测试很有用，但不受生产环境支持。

如果将持久性类型设置为 file，您可以使用目录属性来指定存储 HTTP 会话状态的文件系统位置。保留文件系统对于内部测试很有用，但不受生产环境支持。

- 持久性频率：指定存储会话状态的频率。仅当持久性类型为 ha 时适用。允许的值包括：
  - web-method －将响应发送回客户机之前，将在每个 Web 请求结束时存储会话状态。此模式为发生故障时完全更新会话状态提供了最好的保证。这是默认选项。
  - time-based－在后台按照 reapIntervalSeconds 存储属性设置的频率存储会话状态。此模式不能保证完全更新会话状态。但是，它可以提供很大的性能改善，因为在每个请求之后都不存储状态。要设置此属性，请参见第 200 页上的 “配置存储属性”。
- 持久性范围：指定存储会话状态的量。仅当持久性类型为 ha 时适用。允许的值包括：
  - session －每次都存储整个会话状态。此模式为正确存储任何可分发 Web 应用程序的会话数据提供了最好的保证。这是默认选项。
  - modified-session －如果会话状态已被修改，则存储整个会话状态。如果调用了 HttpSession.setAttribute() 或 HttpSession.removeAttribute()，则系统将认为会话状态已被修改。您必须保证每次更改属性时都调用 setAttribute()。这不是 J2EE 规范的要求，但是此模式需要这样做才能正常工作。
  - modified-attribute －仅存储修改后的会话属性。要使此模式正常工作，您必须遵循一些指导原则。

    每次修改会话状态时都调用 setAttribute()。

    确保各属性之间没有交叉引用。系统将对每个不同的属性关键字的对象图形单独进行序列化并单独存储。如果每个单独的关键字下的对象之间有对象交叉引用，则它们将不会被正确序列化和反序列化。

    在多个属性之间分布会话状态，或者至少在只读属性和可修改属性之间分布会话状态。
- 单点登录状态：选中此框以启用单点登录状态的持久性。要禁用持久性，请取消选中该框。
- HTTP 会话存储：如果更改了用于连接 HADB 的 JDBC 资源以获得会话持久性，您可以更改 HTTP 会话存储。有关详细信息，请参见 Reference Manual 中的 configure-ha-cluster 命令的描述。

2. 单击 “保存”按钮。
3. 要更改影响会话持久性的其他可选设置，请参见第 199 页上的 “配置 Web 容器会话”。
4. 展开 “实例”节点。
5. 选择服务器实例。
6. 转至服务器实例页面。
7. 重新启动服务器。

配置 EJB 容器级别的可用性

要启用可用性并为单个有状态会话 Bean (SFSB) 选择要执行检查点操作的方法，请使用 `sun-ejb-jar.xml` 文件中的设置。有关详细信息，请参见 Developer's Guide。

要使用管理控制台来启用或禁用 EJB 容器的可用性，请执行以下步骤：

1. 选择 “EJB 容器可用性”选项卡，然后选中 “可用性服务”框。要禁用可用性，请取消选中该框。您还可以更改以下可选设置：
   - HA 持久性类型：为已启用可用性的 SFSB 指定会话持久性和钝化机制。允许的值包括 `file`（文件系统）和 `ha` (HADB)。对于需要会话持久性的生产环境，请使用默认值 `ha`。
   - SFSB 持久性类型：为**尚未**启用可用性的 SFSB 指定钝化机制。允许的值包括 `file`（默认值）和 `ha`。

     如果将任一持久性类型设置为 `file`， EJB 容器都将指定用于存储已钝化的会话 Bean 状态的文件系统位置。请参见第 201 页上的 “配置常规 EJB 设置”。文件系统的检查点操作对于内部测试很有用，但不受生产环境支持。
   - SFSB 存储池名称：如果更改了用于连接 HADB 的 JDBC 资源以获得会话持久性，您可以更改 SFSB 存储池名称。有关详细信息，请参见 Reference Manual 中的 `configure-ha-cluster` 命令的描述。
2. 单击 “保存”按钮。
3. 展开 “实例”节点。
4. 选择服务器实例。
5. 转至服务器实例页面。
6. 重新启动服务器。

第 8 章

配置 Java 消息服务资源

本章介绍了如何为使用 Java 消息服务 (JMS) API 的应用程序配置资源。它包含以下各节：

关于 JMS 资源

Application Server 中的 JMS 提供者

Application Server 通过将 Sun Java System Message Queue （以前的 Sun ONE Message Queue）集成到 Application Server 中，实现了 Java 消息服务 (JMS) API。对于基本的 JMS API 管理任务，请使用 Application Server 管理控制台。对于高级任务 （包括管理 Message Queue 群集），请使用 *install_dir*`/imq/bin` 目录中提供的工具。

有关管理 Message Queue 的详细信息，请参见 《Sun Java System Message Queue Administration Guide》。

JMS 资源

Java 消息服务 (JMS) API 使用两种被管理对象：

- 连接工厂，允许应用程序以编程方式创建其他 JMS 对象的对象
- 目标，充当消息的系统信息库

这些对象是以管理方式创建的，创建对象的方式特定于每个 JMS 实现。在 Application Server 中，请执行以下任务：

- 通过创建连接工厂资源来创建连接工厂
- 通过创建两个对象来创建目标：
  - 物理目标
  - 引用物理目标的目标资源

JMS 应用程序使用 JNDI API 来访问连接工厂和目标资源。通常， JMS 应用程序至少使用一个连接工厂和一个目标。要了解所需创建的资源，请仔细研究应用程序或向应用程序开发者进行咨询。

连接工厂分为三种类型：

- `QueueConnectionFactory` 对象，用于点对点通信
- `TopicConnectionFactory` 对象，用于发布 - 订阅通信
- `ConnectionFactory` 对象，可用于点对点通信和发布 - 订阅通信；建议将这些对象用于新的应用程序

目标有两种类型：

- `Queue` 对象，用于点对点通信
- `Topic` 对象，用于发布 - 订阅通信

《 The J2EE 1.4 Tutorial 》中有关 JMS 的章节提供了有关这两类通信和 JMS 其他方面的详细信息 （请参见 `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`）。

创建资源的顺序并不重要。

对于 J2EE 应用程序，请在 Application Server 部署描述符中指定连接工厂和目标资源，如下所示：

- 在 `resource-ref` 或 `mdb-connection-factory` 元素中指定连接工厂 JNDI 名称。
- 在消息驱动 Bean 的 `ejb` 元素和 `message-destination` 元素中指定目标资源 JNDI 名称。
- 在 `message-destination-link` 元素中指定物理目标名称，该元素在企业 Bean 部署描述符的 `message-driven` 元素或 `message-destination-ref` 元素内。此外，还应在 `message-destination` 元素中指定该物理目标名称。（`message-destination-ref` 元素替换了在新的应用程序中过时的 `resource-env-ref` 元素。）在 Application Server 部署描述符的 `message-destination` 元素中，将物理目标名称与目标资源名称链接起来。

JMS 资源与连接器资源之间的关系

Application Server 通过使用名为 `jmsra` 的系统资源适配器实现 JMS。用户创建 JMS 资源时，Application Server 会自动创建连接器资源，这些连接器资源将显示在管理控制台树视图的 “连接器”节点下。

对于用户创建的每个 JMS 连接工厂， Application Server 都将为其创建连接器连接池和连接器资源。对于用户创建的每个 JMS 目标， Application Server 都将为其创建管理对象资源。用户删除 JMS 资源时， Application Server 将自动删除连接器资源。

可以通过使用管理控制台的 “连接器”节点（而不是使用 “JMS 资源”节点）来为 JMS 系统资源适配器创建连接器资源。有关详细信息，请参见第 11 章 连接器资源。

用于 JMS 连接工厂的管理控制台任务

创建 JMS 连接工厂资源

要创建 JMS 连接工厂资源，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后展开 “JMS 资源”节点。
2. 选择 “连接工厂”节点。
3. 在 “JMS 连接工厂”页面中，单击 “新建”。将出现 “创建 JMS 连接工厂”页面。
4. 在 “JNDI 名称”字段中，键入连接工厂的名称。例如：

   `jms/ConnectionFactory1`

   建议的做法是使用 JMS 资源的命名子上下文前缀 `jms/`。
5. 从 “类型”下拉式列表中，选择 `javax.jms.ConnectionFactory`、`javax.jms.QueueConnectionFactory` 或 `javax.jms.TopicConnectionFactory`。
6. 选中 “已启用”复选框以在运行时启用该资源。
7. 在 “高级”区域中，更改连接工厂属性所需的值。有关这些属性的详细信息，请参见第 184 页上的 “编辑连接器连接池”中标题为 “连接器连接池的池设置”的表。 Application Server 会将这些属性应用到为连接工厂创建的连接器连接池。

   对于 JMS 连接工厂资源，请指定事务支持值，如下所示：

   - 为可用于事务 （使用事务范围内多个资源的事务，例如，此资源加上 JDBC 资源、连接器资源或其他 JMS 连接工厂资源）的资源指定 `XATransaction`（默认值）。此值提供最大的灵活性。配置为 `XATransaction` 的资源将参与分为两个阶段的提交操作。
   - 对用于事务的资源 （该事务仅使用事务范围内的唯一一种资源）或在使用多个 XA 资源的分布式事务中用作最后一个代理的资源指定 `LocalTransaction`。此值提供更好的性能。配置为 `LocalTransaction` 的资源将不被用于分为两个阶段的提交操作。
   - 对永远无法参与事务的资源指定 `NoTransaction`；此设置在 JMS 应用程序中有一定的使用。

8. 在“其他属性”区域中，为应用程序所需的属性提供值。下表列出可用的属性。

表 8-1 JMS 连接工厂的其他属性

属性名称	说明
ClientId	为长期订户将要使用的连接工厂指定客户机 ID。
AddressList	指定应用程序将与其进行通信的消息代理实例的名称 （和端口号 [可选]）。列表中的每个地址都指定了要连接的主机名 （和端口号和连接服务 [可选]）。例如，该值可能为 earth 或 earth:7677。当消息代理在默认端口 (7676) 以外的端口上运行时，请指定端口号。如果属性设置指定了群集环境中的多个主机和端口，则将使用该列表中的第一个可用主机 （除非将 AddressListBehavior 属性设置为 RANDOM）。 有关详细信息，请参见 《Sun Java System Message Queue Developer's Guide for Java Clients》。 默认值：本地主机和默认端口号 (7676)。客户机将尝试连接至位于本地主机端口 7676 上的代理。
MessageServiceAddressList	与 AddressList 相同。此属性名已过时。请使用 AddressList 代替。
UserName	进入连接工厂的用户名。 默认值：guest
Password	进入连接工厂的密码。 默认值：guest
ReconnectEnabled	如果启用该属性 （值 = true），则指定客户机运行时在丢失连接时尝试重新连接至消息服务器 （或 AddressList 中的地址列表）。 默认值：TRUE
ReconnectAttempts	指定客户机运行时尝试连接 （或重新连接） AddressList 列表中每个地址的次数。到达这个值后，客户机运行时将尝试连接列表中的下一个地址。值 -1 表示重新连接尝试次数没有限制 （客户机运行时将尝试连接至第一个地址，直到连接成功）。 默认值：3
ReconnectInterval	指定重新连接尝试之间的时间间隔 （毫秒）。此属性适用于对 AddressList 中每个地址的尝试，及对该列表中连续地址的尝试。如果该时间间隔太短，则代理将没有时间恢复。如果该时间间隔太长，则重新连接可能会指明这是不可接受的延迟。 默认值：30000

表 8-1 JMS 连接工厂的其他属性 (续)

属性名称	说明
AddressListBehavior	指定按 AddressList 属性中的地址的顺序 (PRIORITY) 还是按随机顺序 (RANDOM) 进行连接尝试。 RANDOM 表示重新连接将从 AddressList 中随机选择一个地址。如果许多客户机尝试使用同一个连接工厂来进行连接，则该值将阻止它们全部连接至同一个地址。 PRIORITY 表示重新连接始终尝试连接 AddressList 中的第一个服务器地址，而仅在第一个代理不可用时才使用其他的地址。 默认值：RANDOM
AddressListIterations	指定客户机运行时建立 （或重新建立）连接时，在 AddressList 中迭代的次数。值 -1 表示尝试次数没有限制。 默认值：3

9. 在 “目标”区域中，执行以下操作：
   a. 从 “可用”列中选择将要部署使用了资源的应用程序的一个或多个目标。可用目标包括可用群集和可用服务器实例，以及默认的服务器实例 server。
   b. 单击 “添加”以将目标移动到 “选定”列中。
10. 单击 “确定”以保存连接工厂。

等效的 asadmin 命令为：create-jms-resource

编辑 JMS 连接工厂资源

要编辑 JMS 连接工厂资源，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后展开 “JMS 资源”节点。
2. 展开 “连接工厂”节点。
3. 选择要编辑的连接工厂。

4. 在 “编辑 JMS 连接工厂” 页面中，可以执行以下任务：
   - 修改 “说明” 字段中的文本。
   - 选中或取消选中 “已启用” 复选框以启用或禁用资源。
   - 更改 “高级” 区域中的属性值。
   - 添加、删除或修改属性。
5. 或者，单击 “目标” 选项卡以转至 “JMS 连接工厂资源目标” 页面。在此页面中，执行以下操作：
   a. 单击 “管理目标” 以打开 “管理资源目标” 页面。

      在此页面中，在 “可用” 列与 “选定” 列之间移动目标。确保将要部署使用了资源的应用程序的目标放入 “选定” 列中。可用目标包括可用群集和可用服务器实例，以及默认的服务器实例 `server`。单击 “确定” 以保存更改。

   b. 选中目标的复选框，然后单击 “启用” 或 “禁用”，以启用或禁用目标的资源。
6. 单击 “保存” 以保存更改。

删除 JMS 连接工厂资源

要删除 JMS 连接工厂资源，请执行以下步骤：

1. 在树组件中，展开 “资源” 节点，然后展开 “JMS 资源” 节点。
2. 选择 “连接工厂” 节点。
3. 在 “JMS 连接工厂” 页面中，选中要删除的连接工厂的名称旁边的复选框。
4. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-jms-resource`

用于 JMS 目标资源的管理控制台任务

创建 JMS 目标资源

要创建 JMS 目标资源，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后展开 “JMS 资源”节点。
2. 选择 “目标资源”节点。
3. 在 “JMS 目标资源”页面中，单击 “新建”。将出现 “创建 JMS 目标资源”页面。
4. 在 “JNDI 名称”字段中，键入资源的名称。例如：

   `jms/Queue`

   建议的做法是使用 JMS 资源的命名子上下文前缀 `jms/`。
5. 从 “类型”下拉式列表中，选择 `javax.jms.Topic` 或 `javax.jms.Queue`。
6. 选中 “已启用”复选框以在运行时启用该资源。
7. 在 “其他属性”区域中，为各个属性提供值。下表列出可用的属性。

表 8-2 JMS 目标资源的其他属性

属性名称	说明
Name	（必需）资源引用的物理目标的名称。
说明	物理目标的说明。

8. 在 “目标”区域中，执行以下操作：
   a. 从 “可用”列中选择将要部署使用了资源的应用程序的一个或多个目标。可用目标包括可用群集和可用服务器实例，以及默认的服务器实例 `server`。
   b. 单击 “添加”以将目标移动到 “选定”列中。
9. 单击 “确定”。

等效的 `asadmin` 命令为：`create-jms-resource`

编辑 JMS 目标资源

要编辑 JMS 目标资源，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后展开 “JMS 资源”节点。
2. 展开 “目标资源”节点。
3. 选择要编辑的目标资源。
4. 在 “编辑 JMS 目标资源”页面中，可以执行以下任务：
   - 更改资源的类型。
   - 修改 “说明”字段中的文本。
   - 选中或取消选中 “已启用”复选框以启用或禁用资源。
   - 添加、删除或修改 Name 或 Description 属性。
5. 单击 “保存”以保存更改。
6. 或者，单击 “目标”选项卡转至 “JMS 目标资源目标”页面。在此页面中，执行以下操作：
   a. 单击 “管理目标”以打开 “管理资源目标”页面。

      在此页面中，在 “可用”列与 “选定”列之间移动目标。确保将要部署使用了资源的应用程序的目标放入 “选定”列中。可用目标包括可用群集和可用服务器实例，以及默认的服务器实例 `server`。单击 “确定”以保存更改。
   b. 选中目标的复选框，然后单击 “启用”或 “禁用”，以启用或禁用目标的资源。

删除 JMS 目标资源

要删除 JMS 目标资源，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后展开 “JMS 资源”节点。
2. 选择 “目标资源”节点。
3. 在 “JMS 目标资源”页面中，选中要删除的目标资源的名称旁边的复选框。
4. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-jms-resource`

用于 JMS 物理目标的管理控制台任务

创建 JMS 物理目标

要进行生产，务必创建物理目标。但是，在开发和测试阶段，不需要执行此步骤。应用程序首次访问目标资源时， Message Queue 会自动创建目标资源的 Name 属性指定的物理目标。该物理目标是临时的，并且将在 Message Queue 配置属性指定的时间段后过期。

要创建 JMS 物理目标，请执行以下步骤：

1. 在树组件中，展开 “配置”节点，然后展开 “Java 消息服务”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。

3. 选择 “物理目标”节点。
4. 在 “物理目标”页面中，单击 “新建”。将出现 “创建物理目标”页面。
5. 在 “物理目标名称”字段中，键入目标的名称 （例如， PhysicalQueue）。
6. 从 “类型”下拉式列表中，选择 topic 或 queue。
7. 在 “其他属性”区域中，单击 “添加属性”以添加属性。下表列出了当前可用的一个属性。

表 8-3 JMS 物理目标的其他属性

属性名称	说明
maxNumActiveConsumers	可以在从队列目标的负载平衡传送中处于活动状态的使用者的最大数目。如果值为 -1，则表示没有数目限制。如果目标是为独立服务器实例创建的，则默认值为 1 ；如果目标是为群集创建的，则默认值为 -1。

要修改此属性的值或要指定其他物理目标属性，请使用 *install_dir*/imq/bin/imqcmd 命令。有关更多信息，请参见 《Sun Java System Message Queue Administration Guide》。

8. 单击 “确定”。

“物理目标”页面显示系统目标，即过期和无法传送的消息被重定向到的队列 mq.sys.dmq。可以为此目标创建目标资源、使用者和浏览器，但不能删除或向其发送消息。

等效的 asadmin 命令为：create-jmsdest

删除 JMS 物理目标

要删除 JMS 物理目标，请执行以下步骤：

1. 在树组件中，展开 “配置”节点，然后展开 “Java 消息服务”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要为将来的实例 （使用 default-config 的副本）配置默认设置，请选择 default-config 节点。

3. 选择 “物理目标”节点。
4. 在 “物理目标”页面中，选中要删除的目标的名称旁边的复选框。
5. 单击 “删除”。

如果尝试删除系统目标 `mq.sys.dmq`，将显示错误消息。

等效的 `asadmin` 命令为：`delete-jmsdest`

用于 JMS 提供者的管理控制台任务

配置 JMS 提供者的常规属性

使用 “JMS 服务”页面配置所有 JMS 连接都使用的属性。请执行以下步骤：

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 选择 “Java 消息服务”节点以打开 “JMS 服务”页面。
4. 要更改 Application Server 在中止启动之前等待 JMS 服务启动的时间，请编辑 “启动超时”字段中的值。在运行缓慢或过载的系统中，在默认超时值 (60) 的基础上增大该值。

5. 从“类型”下拉式列表中：

   - 选择 LOCAL （server-config 配置的默认设置）以访问本地主机上的 JMS 服务。该 JMS 服务由 Application Server 启动和管理。
   - 选择 REMOTE （default-config 配置的默认设置）以访问其他系统或群集上的 JMS 服务。如果选择了 REMOTE，则下一次启动服务器时 JMS 服务不会由 Application Server 启动，而是通过 Message Queue 来启动和管理 JMS 服务，因而必须单独启动 Message Queue 代理。有关启动代理的信息，请参见 《Sun Java System Message Queue Administration Guide》。如果选择此值并使用了远程主机，请按照第 167 页上的 “编辑 JMS 主机”中的说明来指定远程主机的名称。

6. 在“启动变量”字段中，键入变量以自定义 JMS 服务启动。在 *install_dir*/imq/bin/imqbrokerd 命令中使用任何可用的变量。

7. 使用 “重新连接”复选框指定连接丢失时， JMS 服务是否尝试重新连接至消息服务器 （或 AddressList 中的地址列表）。

   默认情况下，启用重新连接。

8. 在 “重新连接时间间隔”字段中，键入重新连接尝试之间的秒数。此设置适用于对 AddressList 中每个地址的尝试，及对该列表中连续地址的尝试。如果该时间间隔太短，则代理将没有时间恢复。如果该时间间隔太长，则重新连接可能会指明这是不可接受的延迟。

   默认值为 60 秒。

9. 在 “重新连接尝试”字段中，键入客户机运行时尝试连接 （或重新连接）列表中每个地址的次数；到达这个值后，客户机运行时将尝试连接列表中的下一个地址。值 -1 表示重新连接尝试次数没有限制 （客户机运行时将尝试连接至第一个地址，直到连接成功）。

   默认值为 3。

10. 从 “默认 JMS 主机”下拉式列表中选择一个主机。默认值为 default_JMS_host。

11. 在 “地址列表行为”下拉式列表中，选择是按 AddressList 中的地址顺序 (priority) 还是按随机顺序 (random) 来尝试连接。

    priority 表示重新连接始终尝试连接 AddressList 中的第一个服务器地址，而仅在第一个代理不可用时才使用其他地址。

    如果许多客户机同时尝试使用同一个连接工厂来进行连接，则应指定 random 以防止它们全部连接至同一个地址。

    默认值为 random。

12. 在 “地址列表重复”字段中，键入 JMS 服务建立 （或重新建立）连接时，在 `AddressList` 中迭代的次数。值 -1 表示尝试次数没有限制。

默认值为 3。

13. 在 “MQ 模式”和 “MQ 服务”字段中，键入 Message Queue 地址模式名称和 MQ 连接服务名称 （如果要使用非默认模式或服务）。消息服务地址的完整语法为

scheme://*address_syntax*

其中， *scheme* 和 *address_syntax* 将在下表中介绍。

MQ 模式和 MQ 服务是下表的前两列中显示的值。

表 8-4 消息服务器地址模式和语法

模式名称	连接服务	说明	地址语法
`mq`	`jms` 和 `ssljms`	MQ 客户机运行时将连接位于指定主机和端口的 MQ 端口映射器。该端口映射器返回动态建立的连接服务端口的列表，然后 MQ 客户机运行时将连接托管指定连接服务的端口。	[*hostName*][:*port*][/*serviceName*] 默认值： *hostName* = `localhost` *port* = `7676` *serviceName* = `jms` 默认值仅适用于 `jms` 连接服务。对于 `ssljms` 连接服务，需要指定所有的变量 示例： `mq:MyHost:7677/ssljms`
`mqtcp`	`jms`	MQ 客户机运行时将与指定的主机和端口建立 TCP 连接 （绕过 MQ 端口映射器）从而建立连接。	*hostName*:*port*/`jms` 示例： `mqtcp:localhost:7676/jms`

表 8-4 消息服务器地址模式和语法 (续)

模式名称	连接服务	说明	地址语法
mqssl	ssljms	MQ 客户机运行时将与指定的主机和端口建立安全 SSL 连接（绕过 MQ 端口映射器）从而建立连接。	*hostName*:*port*/ssljms 示例： mqssl:localhost:7676/ssljms
http	httpjms	MQ 客户机运行时将与指定 URL 处的 MQ 隧道 Servlet 建立 HTTP 连接。（必须配置代理，以访问 MQ 的 Administrator's Guide 中所述的 HTTP 隧道 Servlet。）	*hostName*:*port*/ *contextRoot*/tunnel 如果多个代理实例使用同一个隧道 Servlet，则连接特定代理实例（而不是随机选择的实例）的语法为：http://*hostName*:*port*/*contextRoot*/tunnel?*serverName*=*hostName*:*instanceName*
https	httpsjms	MQ 客户机运行时将与指定的 MQ 隧道 Servlet URL 建立安全 HTTPS 连接。（必须配置代理，以访问 MQ 的 Administrator's Guide 中所述的 HTTPS 隧道 Servlet。）	*hostName*:*port*/ *contextRoot*/tunnel 如果多个代理实例使用同一个隧道 Servlet，则连接特定代理实例（而不是随机选择的实例）的语法为：http://*hostName*:*port*/*contextRoot*/tunnel?*serverName*=*hostName*:*instanceName*

14. 在“其他属性”区域中，单击“添加属性”以添加属性。下表列出了可用的 Message Queue 代理配置属性。

表 8-5 JMS 提供者的其他属性

属性名称	说明
instance-name	指定完整的 Sun Java System Message Queue 代理实例名称。默认值为 imqbroker。
instance-name-suffix	指定要添加到完整的 Sun Java System Message Queue 代理实例名称中的后缀。该后缀与实例名之间以下划线字符 (_) 分隔。例如，如果实例名为 imqbroker，则在附加后缀 xyz 之后，实例名将更改为 imqbroker_xyz。
append-version	如果为 true，则应在完整的 Sun Java System Message Queue 代理实例名称后附加主要和次要版本号，这两个版本号之前都有一个下划线字符 (_)。例如，如果实例名为 imqbroker，则在附加版本号之后，实例名将更改为 imqbroker_8_0。默认值为 False。

15. 单击 “保存”以保存更改，或者单击 “装入默认值”以恢复服务的默认值。

    单击 "Ping" 以查看 JMS 服务是否已启动并正在运行。如果 JMS 服务已启动并正在运行，则将显示 “Ping 成功：JMS 服务正在运行”的消息。

将提供者和主机改为远程系统会使所有 JMS 应用程序都在远程服务器上运行。要在使用本地服务器的同时使用一个或多个远程服务器，请使用 `AddressList` 属性创建连接工厂资源从而创建访问远程服务器的连接。

有关配置 JMS 服务的更多信息，请参见 《Sun Java System Application Server Developer's Guide》。

等效的 `asadmin` 命令为：`jms-ping`

创建 JMS 主机

要创建 JMS 主机，请执行以下步骤：

1. 在树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 展开 “Java 消息服务”节点。
4. 选择 “JMS 主机”节点。
5. 在 “JMS 主机”页面中，单击 “新建”。将显示 “创建 JMS 主机”页面。
6. 在 “名称”字段中，键入主机的名称。例如：

   ```
   NewJmsHost
   ```

7. 在 “主机”字段中，键入运行 JMS 主机的系统的名称或 Internet 协议 (IP) 地址（`localhost` 或者本地或远程系统的名称）。
8. 在 “端口”字段中，键入 JMS 服务的端口号。仅当要使用的 JMS 服务在非默认端口上运行时，才需要更改此字段。（默认端口为 7676。）

9. 在 “管理用户名”和 “管理密码”字段中，键入 MQ 代理用户名和密码。该用户名和密码与 Application Server 的用户名和密码不同。仅当已使用 *install_dir*/imq/bin/imqusermgr 命令更改了 MQ 代理值时，才需要编辑这些字段。默认值为 admin 和 admin。
10. 单击 “确定”。

等效的 asadmin 命令为：create-jms-host

编辑 JMS 主机

要编辑 JMS 主机，请执行以下步骤：

1. 在树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要为将来的实例 （使用 default-config 的副本）配置默认设置，请选择 default-config 节点。
3. 展开 “Java 消息服务”节点。
4. 选择 “JMS 主机”节点。
5. 在 “JMS 主机”页面中，选择要编辑的主机。
6. 在 “编辑 JMS 主机”页面中，可以执行以下任务：
   - 在 “主机”字段中，更改主机名或 Internet 协议 (IP) 地址。
   - 在 “端口”字段中，更改 JMS 服务的端口号。
   - 更改 “管理用户名”和 “管理密码”字段中的值。
7. 单击 “保存”以保存更改，或者单击 “装入默认值”以恢复主机的默认值。

删除 JMS 主机

要删除 JMS 主机，请执行以下步骤：

1. 在树组件中，展开 “配置” 节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 展开 “Java 消息服务” 节点。
4. 选择 “JMS 主机” 节点。
5. 在 “JMS 主机” 页面中，选中要删除的主机名称旁边的复选框。
6. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-jms-host`

第 9 章

配置 JavaMail 资源

本章介绍了如何为使用 JavaMail API 的应用程序配置资源。它包含以下各节：

关于 JavaMail

JavaMail API

JavaMail API 是一组用于建立邮件系统模型的抽象 API。 API 提供了一个与平台无关以及与协议无关的框架来建立邮件应用程序和消息传送应用程序。 JavaMail API 提供了多个工具用于读取和发送电子邮件。服务提供者可实现特定协议。

JavaMail API 可作为 Java 平台可选软件包实现，还可作为 J2EE 平台的一部分使用。

Application Server 包含 JavaMail API 以及 JavaMail 服务提供者，使应用程序组件可以通过 Internet 发送电子邮件通知，以及从 IMAP 和 POP3 邮件服务器读取电子邮件。

有关 JavaMail API 的详细信息，请转至 JavaMail Web 站点 (`http://java.sun.com/products/javamail/`)。

用于 JavaMail 的管理控制台任务

创建 JavaMail 会话

要创建 JavaMai 会话，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后选择 “JavaMail 会话”节点。
2. 在 “JavaMail 会话”页面中，单击 “新建”。将显示 “创建 JavaMail 会话”页面。
3. 在 “JNDI 名称”字段中，键入会话的名称。例如：

   `mail/MySession`

   建议的做法是使用 JavaMail 资源的命名子上下文前缀 `mail/`。
4. 在 “邮件主机”字段中，键入默认邮件服务器的主机名称。如果未提供特定协议的主机属性， Store 和 Transport 对象的连接方法使用该值。名称必须可以解析为实际的主机名。
5. 在 “默认用户”字段中，键入连接至邮件服务器时要提供的用户名。如果未提供特定协议的用户名属性， Store 和 Transport 对象的连接方法使用该值。
6. 在 “默认返回地址”字段中，键入默认用户的电子邮件地址，格式为：*`username@host.domain`*。
7. 如果您不希望此时启用邮件会话，请取消选中 “已启用”复选框。
8. 在 “高级”区域中，仅当已将 Application Server 的邮件提供者重新配置为使用非默认存储或传输协议时，才需要更改这些字段值。默认情况下，存储协议为 `imap`；存储协议类为 `com.sun.mail.imap.IMAPStore`；传输协议为 `smtp`；传输协议类为 `com.sun.mail.smtp.SMTPTransport`。

   选中 “调试”复选框以启用附加调试输出 （包含此邮件会话的协议跟踪）。如果将 JavaMail 日志级别设置为 `FINE` 或更高，将生成调试输出，并且它将包含在系统日志文件中。有关设置日志级别的信息，请参见第 319 页上的 “配置日志级别”。

9. 在 “其他属性”区域中，单击 “添加属性”以添加应用程序所需的属性 （如特定于协议的主机或用户名属性）。 JavaMail API 文档列出了可用属性 (`http://java.sun.com/products/javamail/javadocs/index.html`)。
10. 在 “目标”区域中，执行以下操作：
    a. 从 “可用”列中选择将要部署使用资源的应用程序的一个或多个目标。可用目标包括可用群集和可用服务器实例，以及默认的服务器实例 `server`。
    b. 单击 “添加”以将目标移动到 “选定”列中。
11. 单击 “确定”以保存会话。

等效的 `asadmin` 命令为：`create-javamail-resource`

编辑 JavaMail 会话

要编辑 JavaMai 会话，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后选择 “JavaMail 会话”节点。
2. 在 “JavaMail 会话”页面中，选择要编辑的会话。
3. 在 “编辑 JavaMail 会话”页面中，您可以执行以下任务：
   - 修改 “邮件主机”、“默认用户”、“默认返回地址”和 “说明”字段中的值。
   - 选中或取消选中 “已启用”复选框以启用或禁用资源。
   - 修改 “高级”字段中的值。
   - 添加、删除或修改属性。
4. 单击 “目标”选项卡以转至 “JavaMail 会话目标”页面。在此页面中，执行以下操作：
   a. 单击 “管理目标”以打开 “管理资源目标”页面。

      在此页面中，在 “可用”列与 “选定”列之间移动目标。确保将要部署使用资源的应用程序的目标添加到 “选定”列中。可用目标包括可用群集和可用服务器实例，以及默认的服务器实例 `server`。单击 “确定”以保存更改。
   b. 选中目标的复选框，然后单击 “启用”或 “禁用”，以启用或禁用目标的资源。
5. 单击 “保存”以保存更改，或单击 “装入默认值”以恢复邮件会话的默认值。

删除 JavaMail 会话

要删除 JavaMai 会话，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后选择 “JavaMail 会话”节点。
2. 在 “JavaMail 会话”页面中，选中要删除的会话的名称旁边的复选框。
3. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-javamail-resource`

第 10 章

JNDI 资源

本章介绍了如何使用管理控制台来配置 JNDI 资源。它包含以下各节：

- 关于 Java 命名和目录接口 (JNDI)
- 关于自定义资源
- 关于外部 JNDI 系统信息库和资源

关于 Java 命名和目录接口 (JNDI)

本节将讨论 Java 命名和目录接口 (JNDI)。 JNDI 是用于访问不同种类的命名和目录服务的应用程序编程接口 (API)。 J2EE 组件通过调用 JNDI 查找方法来定位对象。

本节包括以下主题：

- JNDI 名称和资源
- J2EE 命名服务
- 命名引用和绑定信息

JNDI 名称和资源

JNDI 是 Java 命名和目录接口 API 的首字母缩略词。通过对此 API 进行调用，应用程序可以定位资源和其他程序对象。资源是提供到系统 （如数据库服务器和消息传送系统）的连接的程序对象。（JDBC 资源有时被称为数据源。）每个资源对象都是由唯一的友好名称所标识，称为 JNDI 名称。 Application Server 附带的命名和目录服务将资源对象及其 JNDI 名称绑定在一起。要创建新资源，需要将新的名称 - 对象绑定输入到 JNDI 中。

J2EE 命名服务

JNDI 名称是便于用户使用的对象名称。这些名称通过 J2EE 服务器提供的命名和目录服务绑定到其对象。由于 J2EE 组件通过 JNDI API 访问此服务，因此对象通常使用其 JNDI 名称。例如，Pointbase 数据库的 JNDI 名称为 `jdbc/Pointbase`。当 Sun Java System Application Server 启动时，将从配置文件中读取信息，并自动将 JNDI 数据库名称添加到名称空间。

J2EE 应用程序客户机、企业 Bean 以及 Web 组件都需要具有权限，才能访问 JNDI 命名环境。

应用程序组件的命名环境是一种机制，使用它可以在部署或汇编期间自定义应用程序组件的商业逻辑。使用应用程序组件的环境即可对应用程序组件进行自定义，而无需访问或更改应用程序组件的源代码。

J2EE 容器实现 J2EE 应用程序组件的环境，并将该环境作为 JNDI 命名上下文提供给 J2EE 应用程序组件实例。 J2EE 应用程序组件的环境的使用方式如下：

- 应用程序组件的商业方法使用 JNDI 接口访问该环境。应用程序组件提供商在部署描述符中声明应用程序组件需要其运行时环境提供的所有环境项。
- 容器实现存储应用程序组件环境的 JNDI 命名上下文。容器还提供了部署者可以用于创建和管理每个应用程序组件的环境的工具。
- 部署者使用容器提供的工具，可以初始化应用程序组件的部署描述符中声明的环境项。部署者可以设置和修改环境条目的值。
- 容器使环境命名上下文在运行时可用于应用程序组件实例。应用程序组件的实例使用 JNDI 接口获取环境项的值。

每个应用程序组件定义了其本身的环境项集合。一个应用程序组件在同一容器内的所有实例共享相同的环境项。不允许应用程序组件实例在运行时修改环境。

命名引用和绑定信息

资源引用是部署描述符中的一种元素，用于标识该资源的组件的编码名称。更具体地说，编码名称引用资源的连接工厂。在下节给出的示例中，资源引用名称是 `jdbc/SavingsAccountDB`。

资源的 JNDI 名称与资源引用的名称是不同的。使用此命名方法，您需要在进行部署之前先映射这两个名称，但此方法也用于将组件与资源分离开。由于具有此分离功能，因此如果组件在以后需要访问其他资源，则无需更改名称。这一灵活性使您可以更加容易地从先前存在的组件汇编 J2EE 应用程序。

表 10-1 列出了用于 Sun Java System Application Server 所使用的 J2EE 资源的 JNDI 查找及其关联的引用。

表 10-1 JNDI 查找及其关联的引用

JNDI 查找名称	关联的引用
`java:comp/env`	应用程序环境项
`java:comp/env/jdbc`	JDBC 数据源资源管理器连接工厂
`java:comp/env/ejb`	EJB 引用
`java:comp/UserTransaction`	UserTransaction 引用
`java:comp/env/mail`	JavaMail 会话连接工厂
`java:comp/env/url`	URL 连接工厂
`java:comp/env/jms`	JMS 连接工厂和目标
`java:comp/ORB`	应用程序组件之间共享的 ORB 实例

关于自定义资源

使用自定义资源

自定义资源访问本地 JNDI 系统信息库，外部资源访问外部 JNDI 系统信息库。这两种类型的资源都需要用户指定的工厂类元素、JNDI 名称属性等。在本节中，我们将讨论如何为 J2EE 资源配置 JNDI 连接工厂资源，以及如何访问这些资源。

在 Application Server 中，您可以创建、删除和列出资源以及 list-jndi-entities。

创建自定义资源

要创建自定义资源，请执行以下步骤：

1. 在 Admin Console 的左侧窗格中，为要修改的 JNDI 配置打开 Sun Java System Application Server 实例。
2. 打开 “JNDI”标签，并单击 “自定义资源”。任何已经创建的自定义资源会在右侧窗格中列出。要创建新的自定义资源，单击 “新建”。打开 “JNDI”选项卡，并单击 “新建”。将显示一个页面，用于添加新的自定义资源。
3. 在 “JNDI 名称”字段中，输入用于访问资源的名称。此名称将注册到 JNDI 命名服务中。
4. 在 “资源类型”字段中，输入全限定的类型定义，如上面示例中所示。“资源类型”定义遵循的格式为 `xxx.xxx`。
5. 在 “工厂类”字段中，为要创建的自定义资源输入工厂类名称。该字段中的值是用户指定的工厂类的名称。此类实现了 `javax.naming.spi.ObjectFactory` 接口。
6. 在 “说明”字段中，为要创建的资源输入说明。此说明是字符串值，最多可以包含 250 个字符。
7. 在 “其他属性”部分，添加属性名称和值。
8. 标记 “已启用自定义资源”复选框，以启用自定义资源。
9. 单击 “OK”保存自定义资源。

如果已在群集或独立实例上部署自定义资源，则可以使用 “目标”选项卡管理目标。此选项卡将在创建自定义资源之后显示。通过输入目标名称并单击 “确定”来设置目标。

等效的 asadmin 命令为：create-custom-resource。

编辑自定义资源

要编辑自定义资源，请执行以下步骤：

1. 在 Admin Console 的左侧窗格中，为要修改的 JNDI 配置打开 Sun Java System Application Server 实例。
2. 打开 “JNDI”并选择 “自定义资源”。任何已经创建的自定义资源会在右侧窗格中列出。要编辑自定义资源，请在右侧窗格中单击文件名。
3. 编辑 “资源类型”字段、“工厂类”字段或 “说明”字段。
4. 选中 “启用自定义资源”复选框，将启用自定义资源。
5. 单击 “保存”以保存对自定义资源的更改。

删除自定义资源

要删除自定义资源，请执行以下步骤：

1. 在 Admin Console 的左侧窗格中，打开 "JNDI" 选项卡。
2. 单击 “自定义资源”。任何已经创建的自定义资源会在右侧窗格中列出。要删除自定义资源，请单击要删除的资源名称旁边的框。
3. 单击 “删除”。此自定义资源即被删除。

等效的 asadmin 命令为：delete-custom-resource。

列出自定义资源

要列出自定义资源，请键入 `asadmin ist-custom-resources` 命令。例如，要列出主机 plum 上的自定义资源，请键入以下命令：

```
$asadmin list-custom-resource --host plum target6
```

要查看完整的上下文，请键入 `asadmin help list-custom-resources`。

关于外部 JNDI 系统信息库和资源

使用外部 JNDI 系统信息库和资源

通常，在 Sun Java System Application Server 上运行的应用程序需要访问存储在外部 JNDI 系统信息库中的资源。例如，一般的 Java 对象可能会以 Java 模式存储在 LDAP 服务器中。外部 JNDI 资源元素允许用户配置此类外部资源系统信息库。外部 JNDI 工厂必须实现 `javax.naming.spi.InitialContextFactory` 接口。

使用外部 JNDI 资源的示例：

```
<resources>
<!-- external-jndi-resource 元素指定如何访问存储在外部
-- JNDI 系统信息库中的 J2EE 资源。下面的示例
-- 说明如何访问 LDAP 中存储的 Java 对象。
-- factory-class 元素指定了访问资源工厂时所需的
-- JNDI InitialContext 工厂。属性元素
-- 与适用于外部 JNDI 上下文的环境相匹配，
-- jndi-lookup-name 引用 JNDI 名称，查找并获取
-- 指定的（在此示例中为 java）对象。
-->
<external-jndi-resource jndi-name="test/myBean"
jndi-lookup-name="cn=myBean"
res-type="test.myBean"
factory-class="com.sun.jndi.ldap.LdapCtxFactory">

<property name="PROVIDER-URL"
value="ldap://ldapserver:389/o=myObjects" />
<property name="SECURITY_AUTHENTICATION" value="simple" />
<property name="SECURITY_PRINCIPAL", value="cn=joeSmith,
o=Engineering" />
<property name="SECURITY_CREDENTIALS" value="changeit" />
</external-jndi-resource>
</resources>
```

创建外部资源

要创建外部资源，请执行以下步骤：

1. 在 Admin Console 的左侧窗格中，为要修改的 JNDI 配置打开 Sun Java System Application Server 实例。
2. 打开 “JNDI”并选择 “外部资源”。任何已经创建的外部资源会在右侧窗格中列出。要创建新的外部资源，单击 “New”。
3. 在 “JNDI Name”字段中，输入要访问资源的名称。此名称将登记到 JNDI 命名服务中。
4. 在 “资源类型”字段中，输入全限定的类型定义，如上面示例中所示。“资源类型”定义遵循的格式为 `xxx.xxx`。
5. 在 “JNDI Lookup”字段中，输入要在外部系统信息库中查找的 JNDI 值。例如，在创建一个与外部系统信息库连接的外部资源时，为了测试某个 Bean 类，“JNDI 查找”可能会如下所示：`cn=`*testmybean*。
6. 在 “Factory Class”字段中，输入 JNDI 工厂类外部系统信息库 （例如，`com.sun.jndi.ldap`）。此类实现了 javax.naming.spi. `ObjectFactory` 接口。
7. 在 “说明”字段中，为要创建的资源输入说明。此说明是字符串值，最多可以包含 250 个字符。
8. 在 “其他属性”部分，添加属性名称和值。
9. 标记 “已启用外部资源”复选框，以启用外部资源。
10. 单击 “确定”以保存外部资源。

如果已在群集或独立实例上部署外部资源，则可以使用 “目标”选项卡管理目标。此选项卡将在创建外部资源之后显示。通过输入目标名称并单击 “确定”来设置目标。

等效的 asadmin 命令为：`create-jndi-resource`。

编辑外部资源

要编辑外部资源，请执行以下步骤：

1. 在 Admin Console 的左侧窗格中，为要修改的 JNDI 配置打开 Sun Java System Application Server 实例。
2. 打开 “JNDI” 并选择 “外部资源”。任何已经创建的外部资源会在右侧窗格中列出。要编辑外部资源，请在右侧窗格中单击文件名。
3. 编辑 “资源类型” 字段、“JNDI 查找” 字段、“工厂类” 字段或 “说明” 字段。
4. 选中 “已启用外部资源” 复选框，将启用外部资源。
5. 单击 “保存” 以保存对外部资源的更改。

删除外部资源

要删除外部资源，请执行以下步骤：

1. 在 Admin Console 的左侧窗格中，打开 "JNDI" 选项卡。
2. 单击 “外部资源”。任何已经创建的外部资源会在右侧窗格中列出。要删除外部资源，请单击要删除的资源名称旁边的框。
3. 单击 “删除”。此外部资源即被删除。

等效的 asadmin 命令为 delete-jndi-resource。

列出外部资源

要列出外部资源，请键入 `asadmin list-jndi-resources` 命令并指定 JNDI 名称。例如，键入以下命令可以列出外部资源：

```
$asadmin list-jndi-resources -- target plum jndi_name_test
```

要查看完整的上下文，请键入 `asadmin help list-jndi-resources`。

第 11 章

连接器资源

本章说明如何配置用于访问企业信息系统 (EIS) 的连接器。本章包含以下几节：

关于连接器

连接器模块、连接池和资源

连接器模块也称为资源适配器，是允许应用程序与企业信息系统 (EIS) 进行交互式操作的 J2EE 组件。 EIS 软件包含各种类型的系统：包括企业资源规划 (ERP)、主机事务处理和非关系数据库。类似其他 J2EE 模块，安装连接器模块即是部署该连接器模块。

连接器连接池是一组用于特定 EIS 的可重复使用的连接。要创建连接器连接池，请指定与池关联的连接器模块 （资源适配器）。

连接器资源是为应用程序提供到 EIS 的连接的程序对象。要创建连接器资源，请指定其 JNDI 名称及其关联的连接池。多个连接器资源可以指定一个连接池。应用程序可通过查找资源的 JNDI 名称定位资源。（有关 JNDI 的更多信息，请参见“JNDI 名称和资源”部分）。 EIS 的连接器资源的 JNDI 名称通常位于 `java:comp/env/`*`eis-specific`* 子上下文中。

Application Server 使用连接器模块 （资源适配器）实现 JMS。请参见 “JMS 资源与连接器资源之间的关系”。

连接器连接池任务

- 设置 EIS 访问的一般步骤
- 创建连接器连接池
- 编辑连接器连接池
- 删除连接器连接池

设置 EIS 访问的一般步骤

1. 部署 （安装）连接器。请参见 “部署连接器模块”。
2. 为连接器创建连接池。请参见 “创建连接器连接池”。
3. 创建与连接池关联的连接器资源。请参见 “创建连接器资源”。

创建连接器连接池

创建池之前，请部署与池关联的连接器模块 （资源适配器）。为新池指定的值取决于部署的连接器模块。

要创建连接器连接池，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后展开 “连接器”节点。
2. 选择 “连接器连接池”节点。
3. 在 “连接器连接池”页面中，单击 “新建”。

4. 在 “创建连接器连接池”的第一个页面中，指定以下设置：
   a. 在 “名称”字段中，输入池的逻辑名称。

      创建连接器资源时将指定此名称。

   b. 从 “资源适配器”组合框中选择一个条目。

      该组合框显示已部署的资源适配器 （连接器模块）的列表。
5. 单击 “下一步”。
6. 在 “创建连接器连接池”的第二个页面中，从 “连接定义”组合框中选择一个值。

   组合框中的选项取决于资源适配器。通常，指定一种 ConnectionFactory （工厂实例）以建立与 EIS 的连接。
7. 单击 “下一步”。
8. 在 “创建连接池”的第三个也即最后一个页面中，执行以下任务：
   a. 在 “常规设置”部分中检验各个值是否正确。
   b. 对于 “池设置”部分中的字段，可以保留默认值。

      可以在以后更改这些设置。请参见 “编辑连接器连接池”。
   c. 在 “其他属性”表中添加任何所需的属性。

      在 “创建连接器连接池”的上一个页面中，从 “连接定义”组合框中选择了一个类。如果该类位于服务器的类路径中，则 “其他属性”表将显示默认属性。
9. 单击 “完成”。

等效的 asadmin 命令为：create-connector-connection-pool

- 连接器模块、连接池和资源
- 部署连接器模块

编辑连接器连接池

“编辑连接器连接池”页面使您可以更改池设置和其他属性。

要访问 “编辑连接器连接池”页面，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后展开 “连接器”节点。
2. 展开 “连接器连接池”节点。
3. 选择要编辑的池的节点。
4. 在 “编辑连接器连接池”页面中，您可以更改控制池中连接的数目的设置。请参见表 11-1。

表 11-1 连接器连接池的池设置

参数	说明
初始和最小池大小	池中连接的最小数目。该值还确定了首次创建池或应用程序服务器启动时被置于池中的连接的数目。
最大池大小	池中连接的最大数目。
池大小调整数量	当池向最小池大小方向收缩时，将成批调整大小。此值确定批处理中的连接数目。将该值设置过大会延迟连接回收；而将该值设置过小则会导致效率太低。
空闲超时	连接在池中保持空闲的最长时间 （以秒为单位）。一旦超过此时间，即从池中删除该连接。
最长等待时间	已请求连接的应用程序在达到连接超时之前等待的时间。由于默认等待时间过长，应用程序可能会出现无限期挂起的情况。
一旦失败	选中标记为 “关闭所有连接”的复选框之后，如果单个连接失败，应用程序服务器将关闭池中的所有连接，然后重新建立这些连接。如果未选中此复选框，则只有要使用各个连接时才会重新建立这些连接。

表 11-1 连接器连接池的池设置 (续)

参数	说明
事务支持	使用 “事务支持” 列表可以为连接池选择事务支持类型。选择的事务支持将以向下兼容方式覆盖与此连接池关联的资源适配器中的事务支持属性。也就是说，它可以支持比资源适配器中指定的事务级别低或与其相同的事务级别，但它不能指定更高的级别。 事务支持选项包括以下内容： “事务支持” 菜单中的 “无” 选项表示资源适配器不支持本地资源管理器或 JTA 事务，也不实现 XAResource 或 LocalTransaction 接口。 “本地” 事务支持表示资源适配器将通过实现 LocalTransaction 接口来支持本地事务。本地事务的管理在资源管理器内部进行，不涉及任何外部事务管理器。 "XA" 事务支持表示资源适配器将通过实现 LocalTransaction 和 XAResource 接口来支持本地资源管理器和 JTA 事务。XA 事务由事务管理器在资源管理器外部进行控制和调整。本地事务的管理在资源管理器内部进行，不涉及任何外部事务管理器。

5. 在 “其他属性” 表中，指定名称 - 值对。指定的属性取决于此池使用的资源适配器。部署者使用此表指定的名称 - 值对可用于覆盖由资源适配器供应商定义的属性的默认值。
6. 在 “安全映射” 选项卡窗格中，创建或修改用于连接池的安全映射。有关如何创建安全映射的信息，请参见 “关于安全映射”。
7. 单击 “保存”。

删除连接器连接池

1. 在树组件中，展开 “资源” 节点，然后展开 “连接器” 节点。
2. 选择 “连接器连接池” 节点。
3. 在 “连接器连接池” 页面中，选中要删除的池的复选框。
4. 单击 “删除”。

等效的 asadmin 命令为：delete-connector-connection-pool

连接器资源任务

创建连接器资源

连接器资源 （数据源）为应用程序提供 EIS 连接。创建连接器资源之前，请先创建连接器连接池。

要创建连接器资源，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后展开 “连接器”节点。
2. 展开 “连接器资源”节点。
3. 在 “连接器资源”页面中，单击 “新建”。
4. 在 “创建连接器资源”页面中，指定资源的设置：
   a. 在 “JNDI 名称”字段中，键入唯一的名称，例如：`eis/myERP`。请不要忘记正斜杠。
   b. 从 “池名称”组合框中选择新连接器资源所属的连接池。
   c. 默认情况下，创建资源之后立即可以使用资源 （已启用）。要将资源更改为不可用，请选中 “在所有目标上禁用”单选按钮。
   d. 在此页面的 “目标”部分中，从 “可用”字段中选择连接器资源所在的域、群集或服务器实例，然后单击 “添加”。如果不想将连接器资源部署到 “已选择”字段中列出的某个域、群集或服务器实例，请从字段中选择该实例，然后单击 “删除”。
5. 单击 “确定”。

等效的 `asadmin` 命令为：`create-connector-resource`

编辑连接器资源

1. 在树组件中，展开 “资源”节点，然后展开 “连接器”节点。
2. 展开 “连接器资源”节点。
3. 选择要编辑的连接器资源的节点。
4. 在 “编辑连接器资源”页面中，您可以从 “池名称”菜单中选择其他连接池。
5. 在 “目标”选项卡窗格中，可以通过单击 “管理目标”来编辑连接器资源部署到的目标。有关目标的更多信息，请参见 “创建连接器资源”。
6. 单击 “保存”以应用编辑。

删除连接器资源

1. 在树组件中，展开 “资源”节点，然后展开 “连接器”节点。
2. 选择 “连接器资源”节点。
3. 在 “连接器资源”页面中，选中要删除的资源的复选框。
4. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-connector-resource`

配置连接器服务

使用 “连接器服务”屏幕可以为部署到此群集或服务器实例的所有资源适配器配置连接器容器。

要配置连接器容器，请执行以下步骤：

1. 从树中选择 “配置”。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 "server-config" 节点。
   b. 要为将来的实例 （使用 default-config 的副本）配置默认设置，请选择 "default-config" 节点。

3. 选择 “连接器服务”节点。
4. 在 “关闭超时”字段中指定关闭超时 （以秒为单位）。请输入一个整数，此整数表示应用程序服务器可以等待连接器模块的实例的 `ResourceAdapter.stop` 方法完成的秒数。应用程序服务器将忽略需要比指定关闭超时的时间更长的时间的资源适配器，并且关闭过程将继续。默认的关闭超时为 30 秒。单击 “装入默认值”，可以为部署到此群集或服务器实例的资源适配器选择默认关闭超时。

管理的对象资源任务

创建管理的对象资源

封装在资源适配器 （连接器模块）中的管理的对象为应用程序提供了专用功能。例如，管理的对象可以提供对特定于资源适配器及其关联的 EIS 的分析器的访问。对象可以被管理，即对象可以由管理员配置。要配置对象，请在 “创建管理的对象资源”页面或 “编辑管理的对象资源”页面中添加名称 - 值属性对。创建管理的对象资源时，请使管理的对象与 JNDI 名称相关联。

Application Server 使用资源适配器实现 JMS。对于创建的每个 JMS 目标，Application Server 都会自动创建一个管理的对象资源。

要创建管理的对象资源，请执行以下步骤：

1. 在树组件中，展开 “资源”节点，然后展开 “连接器”节点。
2. 展开 “管理的对象资源”节点。
3. 在 “管理的对象资源”页面中，单击 “新建”。

4. 在 “管理的对象资源”页面中，指定以下设置：
   a. 在 “JNDI 名称”字段中，键入标识资源的唯一名称。
   b. 在 “资源类型”字段中，输入资源的 Java 类型。
   c. 从 “资源适配器”组合框中，选择包含管理的对象的资源适配器。
   d. 选中或取消选中复选框以启用或禁用资源。
   e. 单击 “下一步”。
5. 在 “创建管理的对象资源”的第二个页面中，可以执行以下任务。
   a. 要使用名称 - 值属性对配置管理的对象，请单击 “添加属性”。
   b. 在此页面的 “目标”部分中，从 “可用”字段中选择管理的对象所在的域、群集或服务器实例，然后单击 “添加”。如果不想将管理的对象部署到 “已选择”字段中列出的某个域、群集或服务器实例，请从字段中选择该实例，然后单击 “删除”。
6. 单击 “完成”。

等效的 `asadmin` 命令为：`create-admin-object`

编辑管理的对象资源

1. 在树组件中，展开 “资源”节点，然后展开 “连接器”节点。
2. 展开 “管理的对象资源”节点。
3. 选择要编辑的管理的对象资源的节点。
4. 在 “编辑管理的对象资源”页面中，修改在 “创建管理的对象资源”中指定的值。
5. 在 “目标”选项卡窗格中，通过单击 “管理目标”来编辑管理的对象部署到的目标。有关目标的更多信息，请参见 “创建管理的对象资源”。
6. 单击 “保存”以应用编辑。

删除管理的对象资源

1. 在树组件中，展开 “资源”节点，然后展开 “连接器”节点。
2. 选择 “管理的对象资源”节点。
3. 在 “管理的对象资源”页面中，选中要删除的资源的复选框。
4. 单击 “删除”。

等效的 asadmin 命令为：delete-admin-object

第 12 章

管理命名配置

本章介绍了在 Application Server 中添加、更改以及使用命名的服务器配置。它包含以下各节：

关于命名配置

命名配置

命名配置是一组服务器配置信息。此信息包括针对以下内容的配置设置：HTTP 侦听器、 orb/iiop 侦听器、 JMS 代理、 EJB 容器、安全性、日志记录和监视功能。命名配置中未定义应用程序和资源。

配置是在管理域中创建的。该域中的多个服务器实例或群集可以引用相同的配置，也可以有各自独立的配置。

对于群集，群集中的所有服务器实例都继承群集的配置，从而确保群集实例具有同质环境。

由于命名配置包含如此多的必需配置设置，因此请通过复制现有命名配置来创建新配置。更改新建配置的配置设置之前，该配置与被复制的配置完全相同。

根据群集或实例使用配置的方式的不同，可以将群集和实例分为三类：

- **独立**。独立服务器实例或群集不与其他服务器实例共享其配置；也就是说，其他服务器实例或群集不引用独立服务器实例或群集的命名配置。
- **共享**。共享服务器实例或群集与其他服务器实例或群集共享配置；也就是说，多个实例或群集引用相同的命名配置。
- **群集**。群集服务器实例继承群集的配置。

default-config 配置

`default-config` 配置是一种特殊配置，用作创建独立服务器实例或独立群集配置的模板。非群集服务器实例或群集不允许引用 `default-config` 配置；只能对其进行复制以创建新配置。编辑默认配置，以确保从默认配置复制而来的新配置具有正确的初始设置。

创建实例或群集时创建的配置

创建新服务器实例或新群集时，可以执行以下操作之一：

- 引用现有配置。不添加新配置。
- 创建现有配置的副本。添加服务器实例或群集时，将添加新配置。

默认情况下，在创建新群集或实例时，其配置是从 `default-config` 配置中复制的。要从其他配置进行复制，请在创建新实例或群集时指定要复制的配置。

对于服务器实例，新配置的名称为 *instance_name*`-config`。对于群集，新配置的名称为 *cluster-name*`-config`。

唯一端口号和配置

如果同一主机上有多个实例引用相同的配置，则每个实例必须在唯一的端口号上进行侦听。例如，如果两个服务器实例都引用某个命名配置，该配置包含一个位于端口 80 上的 HTTP 侦听器，则端口冲突将阻止其中一个服务器实例启动。更改用于定义端口号 （各个服务器实例在这些端口号上进行侦听）的属性，从而确保各个实例使用唯一的端口。

以下原则适用于端口号设定：

- 各个服务器实例的端口号最初是从配置继承而来的。
- 创建服务器实例时，如果该端口号已经被使用，则请在实例级别上覆盖继承的默认值，以防止发生端口冲突。
- 假定实例正在共享配置。配置具有端口号 n。如果使用相同的配置在计算机上创建新实例，则为新实例指定的端口号为 n+1 （如果此端口号可用）。如果此端口号不可用，将选择 n+1 后下一个可用的端口。
- 如果您更改了配置的端口号，则继承该端口号的服务器实例将自动继承更改后的端口号。
- 如果您更改了实例的端口号，然后又更改了配置的端口号，则实例的端口号将保持不变。

用于命名配置的管理控制台任务

创建命名配置

要创建命名配置，请执行以下步骤：

1. 在树组件中，选择 “配置” 节点。
2. 在 “配置” 页面中，单击 “新建”。
3. 在 “创建配置” 页面中，为配置输入唯一名称。
4. 选择要复制的配置。

   `default-config` 配置是创建独立服务器实例或独立群集时所使用的默认配置。

等效的 `asadmin` 命令为：`copy-config`。

编辑命名配置的属性

要编辑命名配置的属性，请执行以下步骤：

1. 在树组件中，展开 “配置” 节点。
2. 选择命名配置的节点。
3. 在 “配置系统属性” 页面中，选择是否启用动态重新配置。

   如果已启用，则对配置所作的更改将应用于服务器实例，而无需重新启动服务器。

4. 添加属性、更改属性的当前值或删除属性。

   端口是已定义的属性。如果系统中存在多个服务器实例，则它们各自的端口号必须是唯一的。

表 12-1 包含预定义属性及其说明的列表。

表 12-1 命名配置的属性

属性名称	说明
HTTP_LISTENER_PORT	此属性指定 http-listener-1 的端口号。有效值为 1 到 65535。在 UNIX 中，创建在 1 到 1024 端口上进行侦听的套接字要求具有超级用户权限。
HTTP_SSL_LISTENER_PORT	此属性指定 http-listener-2 的端口号。有效值为 1 到 65535。在 UNIX 中，创建在 1 到 1024 端口上进行侦听的套接字要求具有超级用户权限。
IIOP_SSL_LISTENER_PORT	此属性指定称为 SSL 的 IIOP 侦听器侦听 IIOP 连接的 ORB 侦听器端口。
IIOP_LISTENER_PORT	此属性指定 orb-listener-1 侦听 IIOP 连接的 ORB 侦听器端口。
JMX_SYSTEM_CONNECTOR_PORT	此属性指定 JMX 连接器进行侦听的端口号。有效值为 1 到 65535。在 UNIX 中，创建在 1 到 1024 端口上进行侦听的套接字要求具有超级用户权限。
IIOP_SSL_MUTUALAUTH_PORT	此属性指定称为 SSL_MUTUALAUTH 的 IIOP 侦听器侦听 IIOP 连接的 ORB 侦听器端口。

5. 要编辑与配置关联的所有实例的属性的当前值，请单击 “实例值”。

等效的 `asadmin` 命令为：`set`。

编辑引用配置的实例的端口号

每个引用命名配置的实例最初都从该配置继承端口号。由于系统中的端口号必须是唯一的，因而可能需要覆盖继承的端口号。

1. 在树组件中，展开 “配置”节点。
2. 选择命名配置的节点。
3. 在 “配置系统属性”页面中，单击要编辑的端口号旁边的 “实例值”。

   例如，如果单击 SSL-port 属性旁边的 “实例值”，将看到引用该配置的每个服务器实例的 SSL-port 值。
4. 更改端口值，然后单击 “保存”。

等效的 `asadmin` 命令为：`set`。

查看命名配置的目标

要查看命名配置的目标，请执行以下步骤：

1. 在树组件中，展开 “配置”节点。
2. 选择命名配置的节点。

   “配置系统属性”页面将显示使用该配置的所有目标的列表。对于群集配置，这些目标是群集。对于实例配置，这些目标是实例。

删除命名配置

要删除命名配置，请执行以下步骤：

1. 在树组件中，选择 “配置”节点。
2. 在 “配置”页面中，选中要删除的命名配置的复选框。

   不能删除 `default-config` 配置。
3. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-config`。

第 13 章

J2EE 容器

本章说明如何配置服务器中包含的 J2EE 容器。本章包含以下几节：

- 关于 J2EE 容器
- 用于 J2EE 容器的管理控制台任务

关于 J2EE 容器

本节介绍了 Application Server 中包含的 J2EE 容器。

- J2EE 容器的类型
- Web 容器
- EJB 容器

J2EE 容器的类型

J2EE 容器为 J2EE 应用程序组件提供运行时支持。 J2EE 应用程序组件使用容器的协议和方法访问服务器提供的其他应用程序组件和服务。Application Server 提供了应用程序客户机容器、小应用程序容器、 Web 容器和 EJB 容器。有关显示容器的图表，请参见 “Application Server 体系结构”部分。

Web 容器

Web 容器是用于托管 Web 应用程序的 J2EE 容器。Web 容器通过为开发者提供运行 Servlet 和 JavaServer Page (JSP) 的环境，扩展了 Web 服务器的功能。

EJB 容器

企业 Bean（EJB 组件）是包含商业逻辑的 Java 编程语言服务器组件。 EJB 容器提供对企业 Bean 的本地访问和远程访问。

企业 Bean 分为三种类型：会话 Bean、实体 Bean 和消息驱动 Bean。会话 Bean 表示瞬态对象和进程，并且通常由单个客户机使用。实体 Bean 表示持久性数据，通常保留在数据库中。消息驱动 Bean 用于将消息异步传送到应用程序模块和服务中。

容器负责创建企业 Bean、将企业 Bean 绑定到命名服务以使其他应用程序组件可以访问企业 Bean、确保只有授权的客户机才能访问企业 Bean 的方法、将 Bean 的状态保存到持久性存储中、高速缓存 Bean 的状态以及在必要时激活或钝化 Bean。

用于 J2EE 容器的管理控制台任务

配置常规 Web 容器设置

在本版本中，管理控制台中没有用于 Web 容器的容器范围的设置。

配置 Web 容器会话

本节介绍 Web 容器中的 HTTP 会话设置。HTTP 会话是唯一将状态数据写入持久性存储的 Web 会话。

要设置会话超时值，请执行以下步骤：

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 选择 “Web 容器”节点。
4. 单击 “会话属性”选项卡。
5. 在 “会话超时”字段中，输入会话有效的秒数。
6. 单击 “保存”。

配置管理器属性

会话管理器使您可以配置如何创建和销毁会话、存储会话状态的位置以及会话的最大数目。

要更改会话管理器设置，请执行以下步骤：

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 选择 “Web 容器”节点。
4. 单击 “管理器属性”选项卡。
5. 设置 “Reap 时间间隔”的值。

   “Reap 时间间隔”字段是从存储中删除不活动的会话数据之前的秒数。

6. 设置 “最大会话数”的值。

   “最大会话数”字段是允许的最大会话数目。

7. 设置 “会话文件名”的值。

   “会话文件名”字段是包含会话数据的文件。

8. 设置 “会话 ID 生成器类名”的值。

   “会话 ID 生成器类名”字段使您可以指定用于生成唯一的会话 ID 的自定义类。每个服务器实例只允许有一个会话 ID 生成器类，并且群集中的所有实例必须使用同一会话 ID 生成器，以防止会话密钥冲突。

   自定义会话 ID 生成器类必须实现 `com.sun.enterprise.util.uuid.UuidGenerator` 接口：

```
package com.sun.enterprise.util.uuid;

public interface UuidGenerator {

    public String generateUuid();
    public String generateUuid(Object obj);  //obj is the session object

}
```

   类必须位于 Application Server 类路径中。

9. 单击 “保存”。

配置存储属性

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 选择 “Web 容器”节点。
4. 单击 “存储属性”选项卡。
5. 设置 “Reap 时间间隔”的值。

   “Reap 时间间隔”字段是从存储中删除不活动的会话数据之前的秒数。

6. 单击 “保存”。

配置常规 EJB 设置

本节介绍了以下适用于服务器上所有企业 Bean 容器的设置：

- 会话存储位置
- 池设置
- 高速缓存设置

要覆盖每个容器的默认值，请调整企业 Bean 的 `sun-ejb-jar.xml` 文件中的值。有关详细信息，请参见 Application ServerDeveloper's Guide。（有关指向此指南的链接，请参见 “详细信息”。）

会话存储位置

“会话存储位置”字段指定在文件系统上存储钝化 Bean 和持久的 HTTP 会话所在的目录。

钝化 Bean 是已将其状态写入到文件系统上的文件中的企业 Bean。通常，钝化 Bean 已空闲一段时间并且当前未被客户机访问。

与钝化 Bean 类似，持久的 HTTP 会话是已将其状态写入到文件系统上的文件中的各个 Web 会话。

“提交选项”字段用于指定容器如何高速缓存事务之间的钝化实体 Bean 实例。

“选项 B”用于高速缓存事务之间的实体 Bean 实例，并且是默认选项。“选项 C”用于禁用高速缓存。

池设置

容器维护了一个企业 Bean 池，以便在不创建 Bean 来实现性能的情况下响应客户机请求。这些设置仅适用于无状态会话 Bean 和实体 Bean。

如果在使用已部署的企业 Bean 的应用程序中遇到性能问题，创建池或增加现有池维护的 Bean 的数目有助于提高应用程序的性能。

默认情况下，容器维护企业 Bean 池。

要调整容器的企业 Bean 池的配置，请执行以下步骤：

1. 在树组件中，选择 “配置”节点。

2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要配置所有实例的默认设置，请选择 default-config 节点。
3. 选择 “EJB 容器”节点。
4. 在 “池设置”下的 “初始和最小池大小”字段中，输入容器在池中创建的 Bean 的最小数目。
5. 在 “最大池大小”字段中，输入任何时候容器在池中维护的 Bean 的最大数目。
6. 如果 Bean 处于空闲状态的时间超过 “池空闲超时”中指定的时间，在 “池大小调整数量”字段中输入要从池中删除的 Bean 的数目。
7. 在 “池空闲超时”字段中输入在将池中的 Bean 从池中删除之前 Bean 可以保持空闲状态的时间 （以秒为单位）。
8. 单击 “保存”。
9. 重新启动 Application Server。

高速缓存设置

容器为大多数使用过的企业 Bean 维护了企业 Bean 数据高速缓存。这将允许容器更迅速地响应其他应用程序模块对企业 Bean 的数据请求。本节只适用于有状态会话 Bean 和实体 Bean。

被高速缓存的企业 Bean 处于以下三种状态之一：活动、空闲或钝化。活动企业 Bean 是当前正被客户机访问的企业 Bean。空闲企业 Bean 的数据当前保存在高速缓存中，但没有客户机访问 Bean。钝化 Bean 的数据是被临时存储的，如果客户机请求此 Bean，其数据将被读回高速缓存中。

要调整被高速缓存的企业 Bean 的设置，请执行以下步骤：

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要配置所有实例的默认设置，请选择 default-config 节点。

3. 选择 “EJB 容器”节点。
4. 在 “最大高速缓存大小”字段中调整最大高速缓存大小。

   增加要缓存的 Bean 的最大数目，以消除创建和删除 Bean 的系统开销。但是，如果增加高速缓存，服务器将消耗更多内存和资源。请确保操作环境足够用于高速缓存设置。
5. 在 “高速缓存大小调整数量”字段中调整高速缓存大小调整数量。

   达到高速缓存的 Bean 的最大数目之后，容器将从备份存储中删除一些钝化 Bean，默认设置为 32。
6. 在 “高速缓存空闲超时”字段中，调整为实体 Bean 安排的高速缓存清除速率（以秒为单位）。

   如果高速缓存的实体 Bean 在特定时间内一直处于空闲状态，它将被钝化。即将 Bean 的状态写入备份存储。
7. 在 “删除超时”字段中，调整将有状态会话 Bean 从高速缓存或钝化存储中删除之前的时间 （以秒为单位）。
8. 在 “删除选择策略”字段中，配置容器用来删除有状态会话 Bean 的策略。

   容器将根据在 “删除选择策略”字段中设置的策略决定删除哪个有状态会话 Bean。容器可使用三种可能的策略从高速缓存中删除 Bean：

   - 最近不使用 (NRU)
   - 先进先出 (FIFO)
   - 最近最少使用 (LRU)

   NRU 策略删除最近不使用的 Bean。 FIFO 策略删除高速缓存中最早的 Bean。LRU 策略删除最近最少访问的 Bean。默认情况下，容器使用 NRU 策略。

   通常使用 FIFO 策略删除实体 Bean。
9. 单击 “保存”。
10. 重新启动 Application Server。

配置消息驱动 Bean 设置

消息驱动 Bean 的池与 “配置常规 EJB 设置”中介绍的会话 Bean 的池类似。

默认情况下，容器维护消息 Bean 池。

要调整该池的配置，请执行以下步骤：

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 选择 “EJB 容器”节点。
4. 单击 “MDB 设置”选项卡。
5. 在 “池设置”下的 “初始和最小池大小”字段中，输入容器在池中创建的消息 Bean 的最小数目。
6. 在 “最大池大小”字段中，输入任何时候容器在池中维护的 Bean 的最大数目。
7. 在 “池大小调整数量”字段中，输入当 Bean 处于空闲状态的时间超过 “池空闲超时”中指定的时间时要从池中删除的 Bean 的数目。
8. 在 “池空闲超时”字段中，输入在将池中的 Bean 从池中删除之前 Bean 可以保持空闲状态的时间 （以秒为单位）。
9. 单击 “保存”。
10. 重新启动 Application Server。

配置 EJB 计时器服务设置

计时器服务是由企业 Bean 容器提供的用于安排企业 Bean 使用的通知或事件的持久性和事务性通知服务。所有企业 Bean （有状态会话 Bean 除外）均可从计时器服务接收通知。关闭或重新启动服务器时，服务设置的计时器不会被销毁。

配置计时器服务

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 选择 “EJB 容器”节点。
4. 单击 “EJB 计时器服务”选项卡。
5. 在 “最小传送时间间隔”字段中设置最小传送时间间隔 （以毫秒为单位）。最小传送时间间隔是特定计时器的下一个计时器到期之前允许的最小毫秒数。如果此时间间隔设置得过小，可能会导致服务器过载。
6. 在 “最多重新传送次数”字段中，设置计时器服务尝试传送通知的最大次数。
7. 在 “重新传送时间间隔”字段中，设置两次重新传送尝试之间的时间间隔 （以毫秒为单位）。
8. 单击 “保存”。
9. 重新启动 Application Server。

将外部数据库与计时器服务一起使用

默认情况下，计时器服务使用嵌入式数据库存储计时器。

要使用外部数据库存储计时器，请执行以下步骤：

1. 按照第 137 页上的 “创建 JDBC 资源”中所述为数据库设置 JDBC 资源。
2. 在 “计时器数据源”字段中输入资源的 JNDI 名称。
3. 单击 “保存”。
4. 重新启动 Application Server。

在 `<INSTALL_DIR>/lib/install/databases/` 中提供了 PointBase 和 Oracle 的计时器数据库创建文件样例。

第 14 章

配置安全性

本章介绍了一些核心应用程序服务器的安全性概念，并介绍了如何为 Sun Java System Application Server 8.1 2005Q1 配置安全性。本章包括以下主题：

- 关于 Application Server 安全性
- 用于安全性的管理控制台任务
- 有关区域的管理控制台任务
- 有关 JACC 提供者的管理控制台任务
- 有关审计模块的管理控制台任务
- 有关消息安全性的管理控制台任务
- 有关侦听器和 JMX 连接器的管理控制台任务
- 有关连接器连接池的管理控制台任务
- 使用证书和 SSL
- 详细信息

关于 Application Server 安全性

- 安全性概述
- 关于验证和授权
- 了解用户、组、角色和区域
- 证书和 SSL 简介
- 关于防火墙
- 使用管理控制台管理安全性

安全性概述

安全性是有关数据保护的功能：在存储和传输数据时如何防止对数据进行未经授权的访问或破坏。 Application Server 具有基于 J2EE 标准的动态可扩展安全体系结构，内置了多种安全功能，包括密码学、验证和授权以及公共密钥基本结构。Application Server 是基于 Java 安全模型构建的，该安全模型使用沙盒，应用程序可以在沙盒中安全地运行，而不会给系统或用户带来潜在的危险。本节介绍了以下主题：

了解应用程序和系统安全性

从宽泛意义上讲，应用程序安全性有两种：

- 在**程序安全性**中，开发者编写的应用程序代码负责处理安全事务。作为管理员，您对这一安全机制没有任何控制权。由于程序安全性将安全配置硬编码到应用程序中而不是通过 J2EE 容器对其进行管理，因此这种程序安全性的功能常常受到限制。
- 在**声明安全性**中，容器 (Application Server) 通过应用程序的部署描述符处理安全性事务。您可以通过直接编辑部署描述符或使用 `deploytool` 等工具来控制声明安全性。由于可以在完成应用程序开发之后更改部署描述符，因此声明安全性具有更大的灵活性。

除了应用程序安全性以外，还有影响 Application Server 系统中所有应用程序的**系统安全性**。

程序安全性受应用程序开发者的控制，因此本文档不对其进行讨论；声明安全性受应用程序开发者的控制要少一些，本文档中只偶尔涉及到声明安全性。本文档主要针对系统管理员，因此主要讲述了系统安全性。

管理安全性的工具

Application Server 提供了以下用于管理安全性的工具：

- 管理控制台，它是一种基于浏览器的工具，用于配置整个服务器的安全性，管理用户、组和区域以及执行系统范围内的其他安全性任务。有关管理控制台的一般介绍，请参见 “管理工具”。有关使用管理控制台可以执行的安全性任务的概述，请参见 “使用管理控制台管理安全性”。
- asadmin，一个命令行工具，它可以执行管理控制台能够执行的许多任务。您还可以使用 asadmin 执行某些使用管理控制台无法执行的任务。您可以从命令提示符或在脚本中执行 asadmin 命令，以自动执行重复任务。有关 asadmin 的一般介绍，请参见 “管理工具”。
- deploytool，一个图形形式的封装和部署工具，用于编辑应用程序部署描述符，从而控制各个应用程序的安全性。由于 deploytool 主要针对应用程序开发者，因此本文档未对该工具的使用作详细说明。有关使用 deploytool 的说明，请参见此工具的联机帮助以及位于 http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html 上的 《 The J2EE 1.4 Tutorial 》。

Java 2 Platform, Standard Edition (J2SE) 提供了两个用于管理安全性的工具：

- keytool，一个命令行实用程序，用于管理数字证书和密钥对。使用 keytool 可以管理 certificate 区域内的用户。
- policytool，一个图形实用程序，用于管理系统范围的 Java 安全策略。作为管理员，您很少会用到 policytool。

有关使用 keytool、policytool 和其他 Java 安全性工具的更多信息，请参见位于 http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html#security 上的 Java 2 SDK Tools and Utilities。

在 Enterprise Edition 中，还可以使用两个实现网络安全服务 (NSS) 的工具来管理安全性。有关 NSS 的更多信息，请访问 http://www.mozilla.org/projects/security/pki/nss/。管理安全性的工具包括：

- certutil，一个命令行实用程序，用于管理证书和密钥数据库。
- pk12util，一个命令行实用程序，用于以 PKCS12 格式在证书 / 密钥数据库和文件之间导入和导出密钥及证书。

有关使用 certutil、pk12util 和其他 NSS 安全性工具的更多信息，请参见位于 http://www.mozilla.org/projects/security/pki/nss/tools 的 NSS Security Tools。

管理密码安全性

在此版本的 Application Server 中，包含特定域的规范的 domain.xml 文件最初以明文形式包含了 IMQ 代理的密码。domain.xml 文件中包含此密码的元素为 jms-host 元素的 admin-password 属性。由于在安装期间不能更改此密码，因此它不会对安全性产生很大的影响。

不过，您可以使用管理控制台添加用户和资源，并为这些用户和资源指定密码。部分密码将以明文形式写入 domain.xml 文件，例如用于访问数据库的密码。将这些密码以明文形式保存在 domain.xml 文件中可能会破坏安全性。通过执行以下操作步骤，您可以对 domain.xml 中的任何密码进行加密，包括 admin-password 属性或数据库密码：

1. 在 domain.xml 文件所在的目录 （默认情况下，此目录为 *install_dir*/domains/*domain_dir*/config）中，运行以下 asadmin 命令：

   ```
   asadmin create-password-alias <alias-name>
   ```

   例如，

   ```
   asadmin create-password-alias jms-password
   ```

   将显示输入密码提示 （在本例中为 admin）。有关更多信息，请参阅 create-password-alias、 list-password-aliases 和 delete-password-alias 命令的手册页。

2. 删除并替换 domain.xml 中的密码。使用 asadmin set 命令可以完成此操作。用于此目的的 set 命令的示例如下：

   ```
   asadmin set
   server.jms-service.jms-host.default_JMS_host.admin-password=${ALIAS
   =jms-password}
   ```

3. 重新启动相关域的 Application Server。

保护具有编码密码的文件

某些文件包含需要使用文件系统权限进行保护的编码密码。这些文件包括：

- *install_dir*/domains/*domain_dir*/master-password

  此文件包含编码主密码，并且应使用文件系统权限 600 对其进行保护。

- 任何使用 --passwordfile 参数创建的、作为参数传递给 asadmin 的密码文件均应使用文件系统权限 600 进行保护。

更改主密码

主密码 (MP) 是全局性的共享密码。它从不用于验证，也从不会在网络上传输。此密码是整体安全性的要塞点；用户可以选择在需要时手动输入此密码，也可以将其隐藏在文件中。它是系统中最敏感的数据。用户可以通过删除此文件强制系统提示输入 MP。更改主密码后，系统会将其重新保存到主密码密钥库中。

要更改主密码，必须按照以下步骤进行操作：

1. 停止域的 Application Server。使用 `asadmin` 命令 `change-master-password` 提示输入旧密码和新密码，然后对所有依赖条目重新进行加密。例如，

   ```
   asadmin change-master-password>
   请输入主密码 >
   请输入新的主密码 >
   请再次输入新的主密码 >
   ```

2. 重新启动 Application Server。

警告：此时，不能启动正在运行的服务器实例并且不能重新启动运行服务器实例，除非已更改这些实例所对应的节点代理上的 SMP。如果在更改服务器实例的 SMP 之前重新启动了该服务器实例，它将无法启动。

3. 一次停止一个节点代理及与其相关的服务器。再次运行 `asadmin change-master-password` 命令，然后重新启动节点代理及与其相关的服务器。

4. 继续对下一个节点代理执行此过程，直到对所有节点代理均已执行此过程。这样可以完成滚动更改。

更改管理员密码

“管理密码安全性”中讨论了如何对管理员密码进行加密。强烈建议您对管理员密码进行加密。如果要在对管理员密码进行加密之前更改管理员密码，请使用 `asadmin set` 命令。用于此目的的 `set` 命令的示例如下：

```
asadmin set
server.jms-service.jms-host.default_JMS_host.admin-password=new_pwd
```

还可以使用管理控制台更改管理员密码。要使用管理控制台更改管理员密码，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置” 节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 展开 “安全性” 节点。
4. 展开 “区域” 节点。
5. 选择 `admin-realm` 节点。
6. 在 “编辑区域” 页面中，单击 “管理用户” 按钮。
7. 选择名为 `admin` 的用户。
8. 输入新密码并确认密码。
9. 单击 “保存” 以保存新密码，或单击 “关闭” 以关闭页面而不保存新密码。

指定安全职责

将为以下角色指定安全职责：

- 应用程序开发者
- 应用程序部署者
- 系统管理员

应用程序开发者

应用程序开发者负责：

- 为应用程序组件指定角色和基于角色的访问限制。
- 定义应用程序的验证方法并指定受保护的应用程序部分。

应用程序开发者可以使用 `deploytool` 等工具来编辑应用程序部署描述符。《 The J2EE 1.4 Tutorial 》中的 "*Security*" 一章详细讨论了这些安全性任务，您可以从以下 URL 中查看该教程：

`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`

应用程序部署者

应用程序部署者负责：

- 将用户或组 （或两者）映射到安全角色。
- 限制访问组件方法所需的权限，使满足特定部署方案的要求。

应用程序部署者可以使用 `deploytool` 等工具来编辑应用程序部署描述符。《 The J2EE 1.4 Tutorial 》中的 "*Security*" 一章详细讨论了这些安全性任务，您可以从以下 URL 中查看该教程：

`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`

系统管理员

系统管理员负责：

- 配置安全区域。
- 管理用户帐户和组。
- 管理审计日志。
- 管理服务器证书，并配置服务器对安全套接字层 (SSL) 的使用。
- 处理系统范围内的其他安全性功能，例如连接器连接池的安全映射、其他 JACC 提供者等等。

系统管理员使用管理控制台管理服务器安全性设置，使用 `certutil` 管理证书。本文档主要针对系统管理员。

关于验证和授权

验证和授权是应用程序服务器安全性的核心概念。以下主题讨论了与验证和授权相关的内容：

- 验证实体
- 对用户进行授权
- 指定 JACC 提供者
- 审计验证和授权决策
- 配置消息安全性

验证实体

验证是一个实体 （用户、应用程序或组件）用来确定另一个实体是否是其声明的实体的方法。实体使用**安全凭证**对其自身进行验证。凭证可以是一个用户名和密码、一个数字证书或其他凭证。

通常，验证表示用户使用用户名和密码登录到某个应用程序；也可以指 EJB 从服务器请求资源时，提供安全凭证。通常，服务器或应用程序要求客户机进行验证；另外，客户机也可以要求服务器对其自身进行验证。如果验证是双向的，则称为**双向验证**。

当实体尝试对受保护的资源进行访问时，Application Server 将使用为该资源配置的验证机制来决定是否授予访问权。例如，用户可以在 Web 浏览器中输入用户名和密码，如果应用程序顺利完成了对那些凭证的检验，则表示该用户已通过验证。在此会话余下的时间内，该用户将始终与这个经过验证的安全身份相关联。

Application Server 支持四种类型的验证，如表 14-1 所示。应用程序在其部署描述符中指定所使用的验证类型。有关使用 `deploytool` 来配置应用程序的验证方法的更多信息，请参见位于以下 URL 的 《 The J2EE 1.4 Tutorial 》：

`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`

表 14-1 Application Server 验证方法

验证方法	通信协议	说明	用户凭证加密
基本	HTTP （SSL 可选）	使用服务器的内置弹出式登录对话框。	无，除非使用 SSL。
基于表单	HTTP （SSL 可选）	应用程序提供它自己的自定义登录页面和错误页面。	无，除非使用 SSL。
客户机证书	HTTPS （基于 SSL 的 HTTP）	服务器使用公共密钥证书来验证客户机。	SSL

检验单点登录

单点登录允许一个虚拟服务器实例中的多个应用程序共享用户验证状态。使用单点登录，登录到一个应用程序的用户也会隐式登录到需要相同验证信息的其他应用程序。

单点登录以组为基础。其部署描述符定义了相同的**组**并使用相同的验证方法 （基本、表单、摘要或证书）的所有 Web 应用程序均共享单点登录。

对于为 Application Server 定义的虚拟服务器，默认情况下已启用单点登录。有关禁用单点登录的信息，请参见 “配置单点登录 (SSO)”。

对用户进行授权

用户通过验证后，**授权**级别将决定该用户可以执行哪些操作。用户的授权以其**角色**为基础。例如，人力资源应用程序可以授权管理者查看所有雇员的个人信息，但只允许雇员查看自己的个人信息。有关角色的更多信息，请参见“了解用户、组、角色和区域”。

指定 JACC 提供者

JACC （Java 容器授权合同）属于 J2EE 1.4 规范，它为可插拔授权提供者定义了接口。这使得管理员可以设置第三方插件模块来执行授权。

默认情况下，Application Server 提供一个符合 JACC 规范的基于文件的简单授权引擎。还可以指定其他第三方 JACC 提供者。

JACC 提供者使用 Java 验证和授权服务 (JAAS) API。 JAAS 允许服务验证并强制对用户进行访问控制。 JAAS 实现了 Java 技术版本的标准可插拔验证模块 (PAM) 框架。

审计验证和授权决策

Application Server 可以通过**审计模块**提供对所有验证和授权决策的审计跟踪。Application Server 提供了一个默认的审计模块，还提供了自定义审计模块的功能。有关开发自定义审计模块的信息，请参见 Application Server Developer's Guide。有关 Developer’s Guide 的链接，请参见“详细信息”。

配置消息安全性

消息安全性使服务器可以在消息层执行 Web 服务调用和响应的端对端验证。Application Server 使用 SOAP 层上的消息安全性提供者来实现消息安全性。消息安全性提供者提供了请求和响应消息所需的验证类型等信息。支持的验证类型包括：

- 发件人验证，包括用户名和密码验证。
- 内容验证，包括 XML 数字签名。

该版本附带了两个消息安全性提供者。可以为 SOAP 层的验证配置消息安全性提供者。可以配置的提供者包括 `ClientProvider` 和 `ServerProvider`。

以 （可插入）验证模块的形式将消息层安全性支持集成到 Application Server 及其客户机容器中。默认情况下， Application Server 中的消息层安全性处于禁用状态。

可以为整个 Application Server 或者为特定的应用程序或方法配置消息层安全性。“配置消息安全性”讨论了在 Application Server 级别配置消息安全性。Developer’s Guide 的 "*Securing Applications*" 一章中讨论了如何配置应用程序级别的消息安全性。

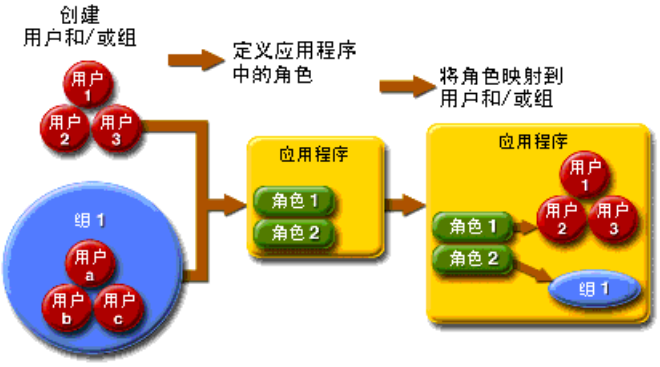
了解用户、组、角色和区域

Application Server 对以下实体强制执行其验证和授权策略：

- 用户：Application Server **中定义的**单个标识。通常，用户是指一个人、一个软件组件（例如企业 Bean），甚至是一种服务。经过验证的用户有时被称为**主体**。用户有时被称为**主题**。
- 组：Application Server **中定义的一组用户**，按照常见特性进行分类。
- 角色：**由应用程序定义的**命名授权级别。可以将角色比喻为开锁的钥匙。许多人都可以有此钥匙的复制钥匙。锁不关心谁要造访，而只关心使用的钥匙是否正确。
- 区域：包含用户和组信息及其关联的安全凭证的系统信息库。区域也被称为**安全策略域**。

注：尽管用户和组是为整个 Application Server 指定的，但是每个应用程序都需要定义自己的角色。当封装和部署应用程序时，应用程序会指定用户 / 组和角色之间的映射，如下图所示。

角色映射



用户

用户是已在 Application Server 中定义的个人（或应用程序）标识。用户可以与组关联。 Application Server 验证服务可以管理多个区域中的用户。

组

J2EE **组**（或简称组）是按常见特性（例如职务或用户概要文件）进行分类的用户类别。例如，假定电子商务应用程序的用户属于 customer 组，但是大客户可以属于 preferred 组。将用户分组可以简化对用户量很大时的访问控制。

角色

角色定义用户可以访问哪些应用程序和每个应用程序的哪些部分以及用户可以执行的操作。也就是说，角色决定了用户的授权级别。

例如，假定在人事应用程序中，所有雇员均可以访问电话号码和电子邮件地址，但只有管理人员才能访问薪水信息。该应用程序至少需要定义两个角色：employee 和 manager；仅允许 manager 角色中的用户查看薪水信息。

角色与用户组的不同之处在于，角色在应用程序中定义功能，而用户组是以某一方式相关的一组用户。例如，假定在人事应用程序中有 full-time、part-time 和 on-leave 几个组，但所有这些组中的用户仍是 employee 角色。

角色是在应用程序部署描述符中定义的。相反，组是针对整个服务器和区域而定义的。应用程序开发者或部署者在每个应用程序的部署描述符中将角色映射到一个或多个组。

区域

区域（也称为**安全策略域**或**安全域**）是服务器定义和强制执行通用安全策略的范围。在实际应用中，区域是服务器存储用户和组信息的系统信息库。

Application Server 预先配置了三个区域：file（初始默认区域）、certificate 和 admin-realm。您还可以设置 ldap、solaris 或自定义区域。应用程序可以在其部署描述符中指定要使用的区域。如果应用程序不指定区域，Application Server 将使用其默认区域。

在 file 区域中，服务器将用户凭证存储在本地名为 keyfile 的文件中。您可以使用管理控制台来管理 file 区域中的用户。有关更多信息，请参见“管理 file 区域用户”。

在 certificate 区域中，服务器将用户凭证存储在证书数据库中。使用 certificate 区域时，服务器结合使用证书和 HTTPS 协议来验证 Web 客户机。有关证书的更多信息，请参见“证书和 SSL 简介”。

admin-realm 也是一个 FileRealm，它将管理员用户凭证存储在本地名为 admin-keyfile 的文件中。您可以使用管理控制台管理此区域中的用户，其方法与您管理 file 区域中的用户的方法相同。有关更多信息，请参见“管理 file 区域用户”。

在 ldap 区域中，服务器将从轻量目录访问协议 (LDAP) 服务器 （例如 Sun Java System Directory Server）中获取用户凭证。 LDAP 是一种协议，它使任何人都可以在网络 （无论是公共 Internet 还是企业内联网）中查找组织、个人和其他资源（例如文件和设备）。有关管理 ldap 区域中的用户和组的信息，请参阅您的 LDAP 服务器文档。

在 solaris 区域中，服务器将从 Solaris 操作系统中获取用户凭证。 Solaris 9 OS 和更高版本支持此区域。有关管理 solaris 区域中的用户和组的信息，请参阅您的 Solaris 文档。

自定义区域是用户凭证的任何其他系统信息库，例如关系型数据库或第三方组件。有关更多信息，请参见 “创建自定义区域”或 Developer's Guide 的 "*Securing Applications*" 一章。

证书和 SSL 简介

本节包括以下主题：

关于数字证书

数字证书 （或简称证书）是在 Internet 上唯一地标识人员和资源的电子文件。证书使两个实体之间能够进行安全、保密的通信。

证书有很多种类型，例如个人证书 （由个人使用）和服务器证书 （用于通过安全套接字层 [SSL] 技术在服务器和客户机之间建立安全会话）。有关 SSL 的更多信息，请参见 “关于安全套接字层”。

证书是基于**公共密钥加密**的，公共密钥加密使用数字**密钥**对 （很长的数字）对信息进行**加密**或编码，从而使信息只能被目标收件人读取。然后，收件人对信息进行**解密** （解码）即可读取该信息。

一个密钥对包含一个公共密钥和一个专用密钥。拥有者对公共密钥进行分发并使任何人都可以使用该公共密钥。但是拥有者永远不会分发专用密钥；专用密钥始终是保密的。由于密钥与数学相关，因此使用了密钥对中的一个密钥进行加密的数据只能通过密钥对中的另一个密钥进行解密。

证书就好像一本护照：它可以标识持有者并提供其他重要信息。证书由称为**证书授权机构** (CA) 的受信任的第三方发布。 CA 类似于护照申领办公室：它将验证证书持有者的身份并对证书进行签名，以使他人无法伪造或篡改证书。 CA 对证书进行签名之后，持有者可以提供该证书作为身份证明并建立经过加密的保密通信。

最重要的是，证书会将拥有者的公共密钥绑定到拥有者的标识。与护照将照片绑定到其持有者的个人信息类似，证书将公共密钥绑定到有关其拥有者的信息。

除了公共密钥以外，证书通常还包括以下信息：

- 持有者的姓名和其他标识，例如使用证书的 Web 服务器的 URL 或个人的电子邮件地址。
- 发布证书的 CA 的名称。
- 失效日期。

数字证书受 x.509 格式的技术规范约束。为检验 `certificate` 区域中某个用户的身份，验证服务将使用 X.509 证书的通用名称字段作为主体名称对 X.509 证书进行检验。

关于证书链

Web 浏览器已预先配置了一组浏览器自动信任的 *root* CA 证书。来自其他证书授权机构的所有证书都必须附带**证书链**，以检验这些证书的有效性。证书链是由一系列 CA 发出的证书序列，最终以根 CA 证书结束。

证书最初生成时是一个**自签名**证书。自签名证书是其签发者 （签名者）与主题 （其公共密钥由该证书进行验证的实体）相同的证书。如果拥有者向 CA 发送证书签名请求 (CSR)，然后输入响应，自签名证书将被证书链替换。链的底部是由 CA 发布的、用于验证主题的公共密钥的证书 （回复）。链中的下一个证书是验证 CA 的公共密钥的证书。通常，这是一个自签名证书 （即，来自 CA、用于验证其自身的公共密钥的证书）并且是链中的最后一个证书。

在其他情况下， CA 可能会返回一个证书链。在此情况下，链的底部证书是相同的（由 CA 签发的证书，用于验证密钥条目的公共密钥），但是链中的第二个证书是由其他 CA 签发的证书，用于验证您向其发送了 CSR 的 CA 的公共密钥。然后，链中的下一个证书是用于验证第二个 CA 的密钥的证书，依此类推，直至到达自签名的**根**证书。因此，链中的每个证书 （第一个证书之后的证书）都需要验证链中前一个证书的签名者的公共密钥。

关于安全套接字层

安全套接字层 (SSL) 是用来确保 Internet 通信和事务安全的最常见的标准。 Web 应用程序使用 HTTPS （基于 SSL 的 HTTP）， HTTPS 使用数字证书来确保在服务器和客户机之间进行安全、保密的通信。在 SSL 连接中，客户机和服务器在发送数据之前都要对数据进行加密，然后由收件人对其进行解密。

当 Web 浏览器 （客户机）需要与某个安全站点建立连接时，则会发生 *SSL* **握手**：

- 浏览器将通过网络发送请求安全会话的消息 （通常请求以 `https` 开头而非 `http` 开头的 URL）。
- 服务器通过发送其证书 （包括公共密钥）进行响应。
- 浏览器将检验服务器的证书是否有效，并且该证书是否是由其证书位于浏览器的数据库中的 CA （并且是受信任的 CA）所签发的。它还将检验 CA 证书是否已过期。
- 如果证书有效，浏览器将生成一个一次性的、唯一的**会话密钥**，并使用服务器的公共密钥对该会话密钥进行加密。然后，浏览器将把加密的会话密钥发送给服务器，这样服务器和浏览器都有一份会话密钥。
- 服务器可以使用其专用密钥对消息进行解密，然后恢复会话密钥。

握手之后，即表示客户机已检验了 Web 站点的身份，并且只有该客户机和 Web 服务器拥有会话密钥副本。从现在开始，客户机和服务器便可以使用该会话密钥对彼此间的所有通信进行加密。这样就确保了客户机和服务器之间的通信的安全性。

SSL 标准的最新版本称为 TLS （传输层安全性）。 Application Server 支持安全套接字层 (SSL) 3.0 和传输层安全性 (TLS) 1.0 加密协议。

要使用 SSL， Application Server 必须拥有接受安全连接的每个外部接口或 IP 地址的证书。只有安装了数字证书之后，大多数 Web 服务器的 HTTPS 服务才能够运行。请使用 “生成服务器证书”中说明的过程来设置您的 Web 服务器可以用于 SSL 的数字证书。

关于加密算法

加密算法是用于加密或解密的加密算法。 SSL 和 TLS 协议支持用于服务器和客户机彼此进行验证、传输证书和建立会话密钥的各种加密算法。

某些加密算法比其他加密算法更强大且更安全。客户机和服务器可以支持不同的加密算法套件。从 SSL3 和 TLS 协议中选择加密算法。在安全连接期间，客户机和服务器同意在通信中使用它们均已启用的最强大的加密算法，因此通常需要启用所有加密算法。

使用基于名称的虚拟主机

对安全应用程序使用基于名称的虚拟主机可能会带来问题。这是 SSL 协议本身的设计限制。必须先进行 SSL 握手 （客户机浏览器在这时接受服务器证书），然后才能访问 HTTP 请求。这样，在验证之前就无法确定包含虚拟主机名的请求信息，因此也不能将多个证书指定给单个 IP 地址。

如果单个 IP 地址上的所有虚拟主机都需要通过同一证书的验证，则添加多个虚拟主机将不会影响服务器上正常的 SSL 操作。但是请注意，大多数浏览器会将服务器的域名与证书中列出的域名 （如果有的话，也主要适用于官方的 CA 签名证书）进行比较。如果域名不匹配，这些浏览器将显示警告。通常在生产环境中，只将基于地址的虚拟主机与 SSL 一起使用。

关于防火墙

防火墙控制两个或多个网络之间的数据流，并管理网络之间的链接。防火墙可以包含硬件和软件元素。本节介绍了一些常用的防火墙体系结构及其配置。此处的信息主要是针对 Application Server 的。有关特定防火墙技术的详细信息，请参阅防火墙供应商提供的文档。

通常，需要对防火墙进行配置，以便客户机访问所需的 TCP/IP 端口。例如，如果 HTTP 侦听器正在端口 8080 上运行，则将防火墙配置为仅允许处理端口 8080 上的 HTTP 请求。同样地，如果为端口 8181 设置了 HTTPS 请求，则必须将防火墙配置为允许处理端口 8181 上的 HTTPS 请求。

如果需要通过 Internet 对 EJB 模块进行直接的 RMI-IIOP 访问， RMI-IIOP 全称为 Remote Method Invocations over Internet Inter-ORB Protocol （通过基于 Internet 的 ORB 间协议的远程方法调用），则还需要打开 RMI-IIOP 侦听器端口，但强烈建议您不要这样做，因为这样可能会破坏安全性。

在双防火墙体系结构中，您必须将外部防火墙配置为允许处理 HTTP 和 HTTPS 事务。您必须将内部防火墙配置为允许 HTTP 服务器插件与防火墙后面的 Application Server 进行通信。

使用管理控制台管理安全性

管理控制台提供了对安全性的以下方面进行管理的方法：

- 服务器安全性设置
- 区域和 file 区域用户
- JACC 提供者
- 审计模块
- 消息安全性
- HTTP 和 IIOP 侦听器安全性
- 管理服务安全性
- 安全映射

服务器安全性设置

在 “安全性设置” 页面中，设置整个服务器的属性，包括指定默认区域、匿名角色和默认的主体用户名和密码。有关更多信息，请参见 “配置安全性设置”。

区域和 file 区域用户

“了解用户、组、角色和区域”中介绍了区域的概念。使用管理控制台可以执行以下任务：

- 创建新区域
- 删除现有区域
- 修改现有区域的配置
- 添加、修改和删除 `file` 区域中的用户
- 设置默认区域

有关这些任务的详细信息，请参见 “有关区域的管理控制台任务”。

JACC 提供者

“指定 JACC 提供者”中介绍了 JACC 提供者。使用管理控制台可以执行以下任务：

- 添加新的 JACC 提供者
- 删除或修改现有 JACC 提供者

有关这些任务的详细信息，请参见 “有关 JACC 提供者的管理控制台任务”。

审计模块

“审计验证和授权决策”中介绍了审计模块。审计是记录重要事件 （例如错误或安全漏洞）以便随后进行检查的方法。所有验证事件都被记录到 Application Server 日志中。完整的访问日志提供了 Application Server 访问事件的顺序线索。

使用管理控制台可以执行以下任务：

- 添加新的审计模块
- 删除或修改现有审计模块

有关这些任务的详细信息，请参见 “有关审计模块的管理控制台任务”。

消息安全性

“配置消息安全性”中介绍了消息安全性的概念。使用管理控制台可以执行以下任务：

- 启用消息安全性
- 配置消息安全性提供者
- 删除或配置现有消息安全性配置或提供者

有关这些任务的详细信息，请参见 “配置消息安全性”。

HTTP 和 IIOP 侦听器安全性

HTTP 服务中的每个虚拟服务器都通过一个或多个 *HTTP* **侦听器**提供网络连接。有关 HTTP 服务和 HTTP 侦听器的常规信息，请参见 “什么是 HTTP 服务？”。

Application Server 支持 CORBA （公共对象请求代理体系结构）对象，这类对象使用 Internet Inter-Orb Protocol (IIOP) 在网络上进行通信。*IIOP* **侦听器**接受来自 EJB 的远程客户机和其他基于 CORBA 的客户机的外来连接。有关 IIOP 侦听器的常规信息，请参见 “IIOP 侦听器”。

使用管理控制台可以执行以下任务：

- 创建新的 HTTP 或 IIOP 侦听器，并指定该侦听器所使用的安全性。
- 修改现有 HTTP 或 IIOP 侦听器的安全性设置。

有关这些任务的详细信息，请参见 “有关侦听器和 JMX 连接器的管理控制台任务”。

管理服务安全性

管理服务决定服务器实例是一个常规实例、一个域管理服务器 (DAS) 还是一个组合。使用管理服务可以配置 JSR-160 兼容的远程 JMX 连接器，该连接器处理域管理服务器与用于远程服务器实例的节点代理 （管理主机上的服务器实例）之间的通信。

使用管理控制台可以执行以下任务：

- 管理管理服务
- 编辑 JMX 连接器
- 修改 JMX 连接器的安全性设置

有关这些任务的详细信息，请参见 “配置管理服务的 JMX 连接器的安全性”。

安全映射

“关于安全映射”中介绍了用于连接器连接池的安全映射的概念。使用管理控制台可以执行以下任务：

- 将安全映射添加到现有连接器连接池中
- 删除或配置现有安全映射

有关这些任务的详细信息，请参见 “有关连接器连接池的管理控制台任务”。

用于安全性的管理控制台任务

配置安全性设置

管理控制台中的 “安全性” 页面允许您设置各种系统范围内的安全性设置。

要编辑这些设置，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置” 节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 选择 “安全性” 节点。

   将显示 “安全性” 页面。
4. 根据需要修改值。表 14-2 说明了常规安全性选项。

表 14-2 常规安全性设置

设置	说明
审计日志记录	选择以启用审计日志记录。如果启用此设置，服务器将装入并运行在 “审计模块” 设置中指定的所有审计模块。如果禁用此设置，服务器将不访问审计模块。默认情况下，此设置处于禁用状态。
默认区域	服务器用来进行验证的活动 （默认）区域。除非应用程序在其部署描述符中指定了其他区域，否则它们将使用该默认区域。列表中将显示所有已配置的区域。初始默认区域为 `file` 区域。
匿名角色	默认或匿名角色的名称。匿名角色将被指定给所有用户。应用程序可以在其部署描述符中使用该角色向任何人授予权限。
默认主体	指定默认用户名。如果未提供任何主体，服务器将使用该默认主体。如果在此字段中输入了一个值，则请在 “默认主体密码” 字段中输入一个相应的值。 正常服务器操作不需要此属性。
默认主体密码	在 “默认主体” 字段中指定的默认主体的密码。 正常服务器操作不需要此属性。
JACC	已配置的 JACC 提供者的类名。有关添加 JACC 提供者的信息，请参见 “创建 JACC 提供者”。
审计模块	审计模块提供者类的列表 （以逗号分隔）。此处所列出的模块必须是已配置模块。如果启用 “审计日志记录”，则此设置必须列出审计模块。默认情况下，服务器使用一个名为 `default` 的审计模块。有关创建新的审计模块的信息，请参见 “创建审计模块”。

5. 在 “其他属性”部分输入要传递给 Java 虚拟机 (JVM) 的其他属性。

   有效属性取决于在 “默认区域”字段中选定的区域的类型。有效属性将在以下各节中进行讨论：

   - 编辑 file 和 admin-realm 区域
   - 编辑 certificate 区域
   - 创建 solaris 区域
   - 创建 ldap 区域
   - 创建自定义区域

6. 选择 “保存”以保存更改，或者选择 “装入默认值”以恢复默认值。

控制对管理工具的访问

只有 asadmin 组中的用户才能访问管理控制台和 asadmin 命令行实用程序。

要为用户授予访问这些管理工具的权限，请将其添加到 admin-realm 的 asadmin 组中。要完成此操作，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 server，请展开 server-config 节点。
   b. 要为所有实例配置默认设置，请展开 default-config 节点。
3. 展开 “安全性”节点。
4. 展开 “区域”节点。
5. 选择 admin-realm 节点。

6. 在 “编辑区域”页面中，单击 “管理用户”按钮。

   初次完成安装之后，安装过程中输入的管理员用户名和密码将列在名为 `admin-keyfile` 的文件中。默认情况下，此用户属于 `asadmin` 组，该组可以授予修改 Application Server 的权限。请仅在您要为用户授予 Application Server 的管理员权限时，才将这些用户指定到该组。

   如果您将用户添加到 `admin-realm` 区域中，但没有将用户指定到 `asadmin` 组，则用户信息仍将被写入名为 `admin-keyfile` 的文件，但用户没有访问管理工具或 `file` 区域中的应用程序的权限。

7. 单击 “新建”将新用户添加到 `admin-realm` 区域中。
8. 在 “用户 ID”、“密码”和 “组列表”字段中输入正确信息。要为用户授予修改 Application Server 的权限，请将 `asadmin` 组包含在 “组列表”中。
9. 单击 “确定”将此用户添加到 `admin-realm` 区域中，或单击 “取消”退出而不保存更改。

有关区域的管理控制台任务

- 创建区域
  - 创建 ldap 区域
  - 创建 solaris 区域
  - 创建自定义区域
- 编辑区域
  - 编辑 file 和 admin-realm 区域
  - 使用网络安全服务 (NSS) 管理用户 (Enterprise Edition)
  - 管理 file 区域用户
  - 编辑 certificate 区域
- 删除区域
- 设置默认区域

创建区域

Application Server 预先配置了三个区域：file、 certificate 和 admin-realm。您还可以创建 ldap、 solaris 和自定义区域。通常，一个服务器上会有每种类型的一个区域，但在 Application Server 上有两个 file 区域：file 和 admin-realm。这两个区域类型相同但用于两种不同的目的。您的系统也可以为每个虚拟服务器配备不同的证书数据库。

要创建安全区域，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置” 节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 server，请展开 server-config 节点。
   b. 要为所有实例配置默认设置，请展开 default-config 节点。
3. 展开 “安全性” 节点。
4. 选择 “区域” 节点。
5. 在 “区域” 页面中，单击 “新建”。

   将显示 “创建区域” 页面。
6. 在 “名称” 字段中为区域输入名称。
7. 指定要创建的区域的类名。表 14-3 显示了有效选项：

表 14-3 区域类名的有效值

区域名	类名
file	com.sun.enterprise.security.auth.realm.file.FileRealm
证书	com.sun.enterprise.security.auth.realm.certificate.CertificateRealm
ldap	com.sun.enterprise.security.auth.realm.ldap.LDAPRealm
solaris	com.sun.enterprise.security.auth.realm.solaris.SolarisRealm
custom	登录区域类的名称

8. 添加区域必需的属性和任何需要的可选属性。

   要添加属性，请执行以下步骤：

   a. 单击 “添加属性”。

   b. 在 “名称”字段中，输入属性的名称。

   - 有关 file 区域属性的说明，请参见 “编辑 file 和 admin-realm 区域”。
   - 有关 certificate 区域属性的说明，请参见 “编辑 certificate 区域”。
   - 有关 ldap 区域属性的说明，请参见 “创建 ldap 区域”。
   - 有关 solaris 区域属性的说明，请参见 “创建 solaris 区域”。
   - 有关自定义区域属性的说明，请参见 “创建自定义区域”。

   c. 在 “值”字段中输入属性的值。

9. 单击 “确定”。

等效的 asadmin 命令为：create-auth-realm

创建 ldap 区域

ldap 区域使用来自 LDAP 服务器的信息执行验证。用户信息包括用户名、密码和用户所属的组。要使用 LDAP 区域，必须已在 LDAP 目录中定义了用户和组。

要创建 LDAP 区域，请按照 “创建区域”中的步骤添加新区域，然后添加表 14-4 中所示的属性。

表 14-4 ldap 区域的必需属性

属性名称	说明	值
目录	目录服务器的 LDAP URL。	LDAP URL 的格式为：ldap:// 主机名 : 端口 例如， ldap://myldap.foo.com:389。
base-dn	用户数据位置的基本独特名称 (DN)，由于将执行树范围的搜索，因此该用户数据的位置可以是高于用户数据的任何级别。搜索树越小，性能越好。	搜索的域，例如： dc=siliconvalley£"dc=BayAreaf" dc=sunf"dc=com。
jaas-context	要用于此区域的登录模块类型。	必须为 ldapRealm。

表 14-5 显示了 ldap 区域的可选属性：

表 14-5 ldap 区域的可选属性

属性名称	说明	默认值
search-filter	用于查找用户的搜索过滤器。	uid=%s （%s 可扩展为主题名称）。
group-base-dn	组数据位置的基本 DN。	与 base-dn 相同，但如果需要也可以将其优化。
group-search-filter	用于查找用户的组成员关系的搜索过滤器。	uniquemember=%d （%d 扩展为用户元素 DN）。
group-target	包含组名称条目的 LDAP 属性名。	CN
search-bind-dn	用于向目录验证以执行搜索惆　似鞑檎业目裳 ° DN。对目录的要求只是不允许进行匿名搜索。	
search-bind-password	search-bind-dn 中给定的 DN 的 LDAP 密码。	

示例

例如，假定在 LDAP 目录中定义了一个 LDAP 用户 Joe Java，如下所示：

```
uid=jjava,ou=People,dc=acme,dc=com
uid=jjava
givenName=joe
objectClass=top
objectClass=person
objectClass=organizationalPerson
objectClass=inetorgperson
sn=java
cn=Joe Java
```

创建或编辑 ldap 区域时，您可以使用示例代码输入表 14-6 中所示的值。

表 14-6 ldap 区域示例值

属性名称	属性值
目录	服务器的 LDAP URL，例如：ldap://ldap.acme.com:389
base-dn	ou=People,dc=acme,dc=com。 可以向更高层次进行搜索 （例如 dc=acme， dc=com），但搜索将遍历树的更大部分，从而导致性能降低。
jaas-context	ldapRealm

创建 solaris 区域

solaris 区域从基础 Solaris 用户数据库 （由系统配置确定）获取用户和组信息。solaris 区域调用基础 PAM 基础结构以用于验证。如果已配置的 PAM 模块需要超级用户权限，则域必须以超级用户身份运行才能使用该区域。有关详细信息，请参见安全服务的 Solaris 文档。

solaris 区域有一个必需的属性 jaas-context，该属性指定要使用的登录模块的类型。属性值必须为 solarisRealm。

注：只有 Solaris 9 或更高版本才支持 solaris 区域。

创建自定义区域

除了四个内置区域以外，您还可以创建以其他方式存储用户数据 （例如存储在关系型数据库中）的自定义区域。自定义区域的开发不在本文档的讨论范围之内。有关更多信息，请参见 Application Server Developer's Guide 的 "Securing Applications"“ 一章。

作为管理员，您需要了解的主要事项是：自定义区域是由源自 Java 验证和授权服务 (JAAS) 软件包的类 （称为 LoginModule）实现的。

要配置 Application Server 以使用自定义区域，请执行以下步骤：

1. 按照 “创建区域” 中概述的过程，输入自定义区域的名称和 LoginModule 类的名称。自定义区域可以使用任何具有唯一性的名称，例如 myCustomRealm。
2. 添加表 14-7 所示的属性：

表 14-7 自定义区域的有效属性

属性名称	属性值
jaas-context	LoginModule 类名，例如 simpleCustomRealm
auth-type	区域的说明，例如 “简单示例自定义区域”。

3. 单击 “确定”。

4. 编辑域的登录配置文件 *install_dir*/domains/*domain_name*/config/login.conf，并在文件末尾添加 JAAS LoginModule 的全限定类名，如下所示：

```
realmName {
    fully-qualified-LoginModule-classname required;
};
```

例如，

```
myCustomRealm {
    com.foo.bar.security.customrealm.simpleCustomLoginModule
required;
};
```

5. 将 LoginModule 类和所有依赖类复制到 *install_dir*/domains/*domain_name*/lib/classes 目录中。
6. 如果控制台中显示 “需要重新启动”，请重新启动服务器。
7. 确保已正确装入区域。

   检查 *install_dir*/domains/*domain_name*/logs/server.log 以确保服务器已装入该区域。服务器应调用区域的 init() 方法。

编辑区域

要编辑区域，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 server，请展开 server-config 节点。
   b. 要为所有实例配置默认设置，请展开 default-config 节点。
3. 展开 “安全性”节点。
4. 展开 “区域”节点。
5. 选择一个现有区域的名称。

   将显示 “编辑区域”页面。

6. 根据需要编辑现有属性及其值。

   有关 file 区域属性的信息，请参见“编辑 file 和 admin-realm 区域”。要管理 file 区域中的用户，请单击“管理用户”按钮；有关更多信息，请参见“管理 file 区域用户”。

   有关 certificate 区域属性的信息，请参见“编辑 certificate 区域”。

7. 要添加其他属性，请单击“添加属性”按钮。该页面将显示一个新行。输入一个有效的属性名和属性值。有关可以配置的可选属性的说明，请参见下列各表：

   - 表 14-4， ldap 区域的必需属性
   - 表 14-5， ldap 区域的可选属性
   - 表 14-7，自定义区域的有效属性
   - 表 14-8， file 区域的必需属性
   - 表 14-9， certificate 区域的可选属性

8. 单击“保存”以保存更改。

编辑 file 和 admin-realm 区域

服务器在 file 区域的名为 keyfile 的文件和 admin-realm 区域的名为 admin-keyfile 的文件中维护所有用户、组和密码信息。对于这两种区域， file 属性均指定了 keyfile 的位置。表 14-8 显示了 file 区域的必需属性。

表 14-8 file 区域的必需属性

属性名称	说明	默认值
file	密钥文件的完整路径和名称。	*install_dir*/domains/*domain-name*/config/keyfile
jaas-context	要用于此区域的登录模块类型。	fileRealm 是唯一的有效值

keyfile 最初为空，因此在使用 file 区域之前，必须先添加用户。有关说明请参见“管理 file 区域用户”。

admin-keyfile 最初包含管理员用户名、加密格式的管理员密码和该用户所属的组（默认情况下为 asadmin）。有关将用户添加到 admin-realm 的更多信息，请参见“控制对管理工具的访问”。

注： admin-realm 的 asadmin 组中的用户已被授权，可以使用管理控制台和 asadmin 工具。只能将具有服务器管理权限的用户添加到该组中。

使用网络安全服务 (NSS) 管理用户

仅在 **Enterprise Edition** 中，您可以使用管理控制台来管理用户 （如 “管理 file 区域用户”所述），或者使用 NSS 工具来管理用户。网络安全服务 (NSS) 是为支持启用安全性的客户机和服务器应用程序的跨平台开发而设计的一组库。使用 NSS 构建的应用程序可以支持 SSL v2 和 v3、TLS、PKCS #5、PKCS #7、PKCS #11、PKCS #12、 S/MIME、 X.509 v3 证书和其他安全性标准。有关详细信息，请链接到以下 URL：

- 位于 http://www.mozilla.org/projects/security/pki/nss/ 上的 Network Security Services (NSS)
- 位于 http://www.mozilla.org/projects/security/pki/nss/tools/ 上的 NSS Security Tools
- 位于 http://www.mozilla.org/projects/security/pki/nss/overview.html 上的 Overview of NSS

管理 file 区域用户

使用管理控制台来管理 `file` 区域用户。 `file` 区域中的用户和组列在 keyfile 文件中，该文件的位置由 `file` 属性指定。

注：还可以使用以下步骤将用户添加到任何 file 区域 （包括 `admin-realm`）中。您只需将本节中引用的 `file` 区域替换为目标区域的名称即可完成操作。

`file` 区域中的用户可以属于 *J2EE* **组**，它是按照常见属性分类的一个用户类别。例如，假定电子商务应用程序的用户属于 `CUSTOMER` 组，但是大客户可以属于 `PREFERRED` 组。将用户分组可以简化对用户量很大时的访问控制。

初次安装 Application Server 之后，唯一的用户是管理员在安装过程中输入的用户。默认情况下，此用户属于 `admin-realm` 区域中的 `asadmin` 组，该组可以授予修改 Application Server 的权限。指定给该组的任何用户都将具有管理员权限，也就是说，这些用户具有对 `asadmin` 工具和管理控制台的访问权。

要管理 `file` 区域用户，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。

3. 展开 “安全性” 节点。
4. 展开 “区域” 节点。
5. 选择 file 节点。
6. 在 “编辑区域” 页面中，单击 “管理用户” 按钮。

   将显示 “文件用户” 页面。在此页面中执行以下任务：

   - 添加用户
   - 编辑用户
   - 删除用户

添加用户

在 “文件用户” 页面中，执行以下步骤来添加新用户：

1. 单击 “新建” 将新用户添加到 file 区域中。
2. 在 “文件用户” 页面中输入以下信息：
   - **用户 ID** （**必填**）－用户名。
   - **密码** （**必填**）－用户的密码。
   - **确认密码** （**必填**）－再次输入用户密码进行检验。
   - **组列表** （**可选**）－用户所属的组列表 （以逗号分隔）。无需在其他位置定义这些组。
3. 单击 “确定” 将此用户添加到 file 区域的用户列表中。单击 “取消” 退出而不保存更改。

等效的 asadmin 命令为：create-file-user

编辑用户

在 “文件用户” 页面中，执行以下步骤来更改用户的信息：

1. 在 “用户 ID” 列中，单击要修改的用户名。

   将显示 “编辑文件区域用户” 页面。
2. 在 “密码” 和 “确认密码” 字段中输入新密码来更改用户密码。
3. 在 “组列表” 字段中添加或删除组来更改用户所属的组。用逗号将组名分隔开。不需要先定义组。
4. 单击 “保存” 将此用户保存到 file 区域的用户列表中。单击 “关闭” 退出而不保存更改。

删除用户

在 “文件用户” 页面中，执行以下步骤来删除用户：

1. 选中要删除的用户名左侧的复选框。
2. 单击 “删除”。
3. 单击 “关闭” 返回到 “编辑区域” 页面。

等效的 asadmin 命令为：delete-file-user

编辑 certificate 区域

certificate 区域支持 SSL 验证。该区域在 Application Server 的安全上下文中设置用户身份，并使用从信任存储和密钥库文件中以加密方式检验的客户机证书中获得的用户数据填充该身份 （请参见 “关于证书文件”）。使用 certutil 将用户添加到这些文件中。使用 certificate 区域， J2EE 容器可以根据每个用户从其证书中获得的独特名称 (DN) 来执行授权处理。DN 是证书对其公共密钥进行标识的实体的名称。此名称使用 X.500 标准，因此它在 Internet 中应该是唯一的。有关密钥库和信任存储的更多信息，请参阅 certutil 文档，位于 “关于 CertUtil 实用程序”。

表 14-9 列出了 certificate 区域的可选属性。

表 14-9 certificate 区域的可选属性

属性	说明
assign-groups	以逗号分隔的组名列表。提供有效证书的所有客户机均被指定给这些组。例如， employee,manager，它们是用户组的名称。
jaas-context	要用于此区域的登录模块类型。对于 certificate 区域，该值必须为 certificateRealm。

配置双向验证

- 为所有应用程序启用双向验证
- 在应用程序中启用双向 SSL 验证

在双向验证中，服务器端和客户端验证都会被启用。要测试双向验证，必须存在一个包含有效证书的客户机。有关双向验证的信息，请参见位于以下位置的 《 The J2EE 1.4 Tutorial 》中的 "*Security*" 一章：

http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

为所有应用程序启用双向验证

Application Server 使用 `certificate` 区域进行 HTTPS 验证。

要指定对使用此区域的所有应用程序进行双向验证，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 展开 “安全性”节点。
4. 展开 “区域”节点。
5. 选择 `certificate` 区域。
6. 单击 “添加属性”按钮。
   - 在 “名称”字段中，输入 `clientAuth`。
   - 在 “值”字段中，输入 `true`。
7. 单击 “保存”。
8. 如果控制台中显示 “需要重新启动”，请重新启动 Application Server。

   重新启动服务器之后，需要对使用 `certificate` 区域的所有应用程序进行客户机验证。

在应用程序中启用双向 *SSL* 验证

要启用对特定应用程序的双向验证，请使用 `deploytool` 将验证的方法设置为 `Client-Certificate`。有关使用 `deploytool` 的更多信息，请参阅位于以下位置的《 The J2EE 1.4 Tutorial 》中的 "*Security*" 一章：

`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`。

删除区域

要删除区域，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 展开 “安全性”节点。
4. 选择 “区域”节点。
5. 单击要删除的区域旁边的框。
6. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-auth-realm`

设置默认区域

默认区域是当应用程序的部署描述符未指定区域时 Application Server 用于验证和授权的区域。

要设置默认区域，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 选择 “安全性”节点。

   将显示 “安全性”页面。
4. 在 “默认区域”字段中，从下拉式列表中选择所需的区域。
5. 单击 “保存”以保存更改，或单击 “装入默认值”以删除更改并恢复 Application Server 的默认值。
6. 如果控制台中显示 “需要重新启动”，请重新启动服务器。

有关 JACC 提供者的管理控制台任务

创建 JACC 提供者

JACC（Java 容器授权合同）属于 J2EE 1.4 规范，它为可插拔授权提供者定义了接口。这使得管理员可以设置第三方**插件**模块来执行授权。默认情况下， Application Server 提供一个简单的、 JACC 兼容的、基于文件的授权引擎。

要创建 JACC 提供者，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 展开 “安全性”节点。
4. 选择 “JACC 提供者”节点。
5. 在 “JACC 提供者”页面中，单击 “新建”。
6. 在 “创建 JACC 提供者”页面中，输入以下信息：
   - **名称**－用于标识此提供者的名称。
   - **策略配置**－实现策略配置工厂的类的名称。默认提供者使用 `com.sun.enterprise.security.provider.PolicyConfigurationFactoryImpl`。
   - **策略提供者**－实现策略工厂的类的名称。默认提供者使用 `com.sun.enterprise.security.provider.PolicyWrapper`。

7. 单击 “添加属性” 按钮向提供者添加属性。有效属性包括：
   - repository：包含策略文件的目录。对于默认提供者，此值为 *install_dir*/domains/*domain_dir*/generated/policy。
8. 单击 “确定” 保存此配置，或单击 “取消” 退出而不保存更改。

编辑 JACC 提供者

要编辑 JACC 提供者，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置” 节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 server，请展开 server-config 节点。
   b. 要为所有实例配置默认设置，请展开 default-config 节点。
3. 展开 “安全性” 节点。
4. 展开 “JACC 提供者” 节点。
5. 选择要编辑的 JACC 提供者的节点。
6. 在 “编辑 JACC 提供者” 页面中，根据需要修改提供者信息：
   - **策略配置**－实现策略配置工厂的类的名称。
   - **策略提供者**－实现策略工厂的类的名称。
7. 要添加属性，请单击 “添加” 按钮。输入属性的名称和值。有效条目包括：
   - repository：包含策略文件的目录。对于默认提供者，该值为 ${com.sun.aas.instanceRoot}/generated/policy。
8. 要删除现有属性，请单击属性左侧的复选框，然后单击 “删除属性”。
9. 单击 “保存” 以保存更改，或单击浏览器的 “后退” 按钮以取消而不保存更改。

删除 JACC 提供者

要删除 JACC 提供者，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 展开 “安全性”节点。
4. 选择 “JACC 提供者”节点。
5. 单击要删除的 JACC 提供者左侧的复选框。
6. 单击 “删除”。

设置活动的 JACC 提供者

要指定 JACC 提供者，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 选择 “安全性”节点。

   将显示 “安全性”页面。
4. 在 "JACC" 字段中，输入服务器要使用的 JACC 提供者的名称。

   如果不知道哪些 JACC 提供者可用，请展开树视图中的 “JACC 提供者”组件来查看所有已配置的 JACC 提供者。
5. 选择 “保存”以保存更改，或者选择 “装入默认值”以返回默认值。
6. 如果控制台中显示 “需要重新启动”，请重新启动 Application Server。

有关审计模块的管理控制台任务

创建审计模块

Application Server 提供了一个简单的默认审计模块；有关更多信息，请参见 “使用默认审计模块”。

要创建新的审计模块，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置” 节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 展开 “安全性” 节点。
4. 选择 “审计模块” 节点。
5. 在 “审计模块” 页面中，单击 “新建”。
6. 在 “创建审计模块” 页面中，输入以下信息：
   - **名称** － 用于标识此审计模块的名称。
   - **类名** － 实现此模块的类的全限定名称。默认审计模块的类名为 `com.sun.enterprise.security.Audit`。
7. 要将 JVM 属性添加到此模块中，请单击 “添加属性”。指定每个属性的名称和值。有效属性包括：
   - `auditOn` － 指定是否启用此实现类。有效值为 `true` 和 `false`。
8. 单击 “确定” 以保存条目，或单击 “取消” 以退出而不保存更改。

编辑审计模块

默认情况下，未打开审计模块。有关如何激活审计模块的更多信息，请参阅 “启用和禁用审计日志记录”。

要编辑审计模块，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 server，请展开 server-config 节点。
   b. 要为所有实例配置默认设置，请展开 default-config 节点。
3. 展开 “安全性”节点。
4. 展开 “审计模块”节点。
5. 选择要编辑的审计模块的节点。
6. 在 “编辑审计模块”页面中，根据需要修改类名。
7. 选择 “添加”按钮并输入属性的名称和值来输入模块的所有其他属性。有效属性包括：
   - auditOn —指定是否使用此审计模块。有效值为 true 和 false。
8. 通过选择要修改的名称或值并在文本字段中直接输入更改来修改任何现有属性。
9. 通过选中属性左侧的复选框并单击 “删除属性”来删除属性。
10. 单击 “保存”以保存更改，或单击浏览器上的 “后退”按钮以取消而不保存更改。

删除审计模块

要删除审计模块，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 server，请展开 server-config 节点。
   b. 要为所有实例配置默认设置，请展开 default-config 节点。

3. 展开 “安全性” 节点。
4. 选择 “审计模块” 节点。
5. 单击要删除的审计模块左侧的复选框。
6. 单击 “删除”。

启用和禁用审计日志记录

要指定服务器使用的审计模块，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置” 节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 选择 “安全性” 节点。

   将显示 “安全性” 页面。
4. 要启用日志记录，请选中 “审计日志记录” 复选框。要禁用日志记录，请取消选中该复选框。选择此选项可以装入审计模块并确保这些模块在审计时被 Application Server 的审计库调用。
5. 如果要启用审计日志记录，请指定一个默认审计模块 （如 “设置活动的审计模块”中所述）。
6. 选择 “保存” 以保存更改。
7. 如果控制台中显示 “需要重新启动”，请重新启动 Application Server。

设置活动的审计模块

要指定服务器使用的审计模块，请按照 “启用和禁用审计日志记录”所述启用审计日志记录，然后执行以下步骤：

1. 在 “审计模块”字段中，输入服务器要使用的审计模块的名称。（预配置的审计模块被称为 `default`。）确保已按照 “启用和禁用默认审计模块”所述将该审计模块的 auditOn 设置为 true。
2. 选择 “保存”以保存更改，或选择 “装入默认值”以取消更改。
3. 如果控制台中显示 “需要重新启动”，请重新启动 Application Server。

使用默认审计模块

`default` 审计模块将验证和授权请求记录到服务器日志文件中。有关更改日志文件位置的信息，请参见 “配置常规日志记录设置”。

验证日志条目包括以下信息：

- 已尝试进行验证的用户的名称。
- 处理过访问请求的区域。
- 请求的 Web 模块 URI 或 EJB 组件。
- 请求成功或失败。

无论是否启用审计日志记录， Application Server 都将记录所有被拒绝的验证事件。

授权日志条目包括以下信息：

- 经过验证的用户 （如果有）的名称。
- 请求的 Web URI 或 EJB 组件。
- 请求成功或失败。

启用和禁用默认审计模块

除了启用日志记录以外，还需要设置特定审计模块所需的所有属性。如果是默认审计模块，执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置” 节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 展开 “安全性” 节点。
4. 展开 “审计模块” 节点。
5. 单击 `default` 节点。
6. 将 `auditOn` 属性的值设置为 `true`。
7. 选择 “保存” 以保存更改。
8. 如果控制台中显示 “需要重新启动”，请重新启动 Application Server。

有关侦听器和 JMX 连接器的管理控制台任务

- 配置 HTTP 侦听器的安全性
- 配置 IIOP 侦听器的安全性
- 配置管理服务的 JMX 连接器的安全性
- 设置侦听器安全性属性

配置 HTTP 侦听器的安全性

HTTP 服务中的每个虚拟服务器都通过一个或多个 *HTTP* **侦听器**提供网络连接。使用管理控制台可以创建新的 HTTP 侦听器和编辑现有 HTTP 侦听器的安全性设置。

要编辑现有 HTTP 侦听器的安全性设置，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 展开 “HTTP 服务”节点。
4. 选择 “HTTP 侦听器”节点。
5. 选择 HTTP 侦听器来编辑现有的侦听器，或者单击 “新建”并执行 “创建 HTTP 侦听器”中的步骤来创建新的侦听器。
6. 按照 “设置侦听器安全性属性”中的过程设置安全性属性。
7. 单击 “保存”以保存更改，或单击浏览器的 “后退”按钮以取消而不保存更改。

等效的 `asadmin` 命令为：`create-http-listener`

配置 IIOP 侦听器的安全性

Application Server 支持 CORBA （公共对象请求代理体系结构）对象，这类对象使用 Internet Inter-Orb Protocol (IIOP) 在网络上进行通信。*IIOP* **侦听器**接受来自 EJB 的远程客户机和其他基于 CORBA 的客户机的外来连接。使用管理控制台可以创建新的 IIOP 侦听器和编辑现有 IIOP 侦听器的设置。

要编辑 IIOP 侦听器的安全性属性，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。

2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 server，请展开 server-config 节点。
   b. 要为所有实例配置默认设置，请展开 default-config 节点。
3. 展开 "ORB" 节点。
4. 选择 “IIOP 侦听程序” 节点。
5. 选择一个 IIOP 侦听器来编辑该侦听器，或者单击 “新建” 并执行 “创建 IIOP 侦听器” 中的步骤来创建新的侦听器。
6. 按照 “设置侦听器安全性属性” 中的过程设置安全性属性。
7. 单击 “保存” 以保存更改，或者单击 “装入默认值” 以恢复属性的默认值。

如果创建了新的侦听器，则新侦听器现在将列在 “IIOP 侦听器” 页面的 “当前侦听器” 表中。

等效的 asadmin 命令为：create-iiop-listener

配置管理服务的 JMX 连接器的安全性

要编辑管理服务中的 JMX 连接器的安全性属性，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置” 节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 server，请展开 server-config 节点。
   b. 要为所有实例配置默认设置，请展开 default-config 节点。
3. 展开 “管理服务” 节点。
4. 选择要修改的管理服务。
5. 按照 “设置侦听器安全性属性” 中的过程设置安全性属性。
6. 单击 “保存” 以保存更改，或者单击 “装入默认值” 以恢复属性的默认值。

设置侦听器安全性属性

按照以下设置 HTTP 侦听器、 IIOP 侦听器和 JMX 连接器安全性属性的一般过程进行操作：

1. 在 “编辑 HTTP 侦听器”、“编辑 IIOP 侦听器”或 “编辑 JMX 连接器”页面中，转至标有 “SSL”的部分。
2. 选中 “侦听器”字段中的 “已启用”框来启用该侦听器的安全性。当选中此选项时，您必须选择 SSL3 或 TLS 来指定启用何种类型的安全性，并且必须输入证书昵称。
3. 如果客户机在使用此侦听器时要向 Application Server 验证自身，请在 “客户机验证”字段中选中 “已启用”框。
4. 如果已选中 “已启用”框，请在 “证书昵称”字段中输入密钥库别名。密钥库别名是标识现有服务器密钥对和证书的单个值。默认密钥库的证书昵称为 `slas`。

   要查找证书昵称，请使用 `certutil` 实用程序，如 “关于 CertUtil 实用程序”所述。
5. 如果已选中 “已启用”框，请选择 SSL3 和/或 TLS。默认情况下， SSL3 和 TLS 均处于启用状态。
6. 根据需要启用单个加密算法套件。默认情况下，所有支持的加密算法套件均处于启用状态。加密算法将在 “关于加密算法”中讨论：
7. 选择 “保存”以保存更改，或选择 “装入默认值”以取消更改。

有关虚拟服务器的管理控制台安全性任务

- 配置单点登录 (SSO)

配置单点登录 (SSO)

单点登录允许多个应用程序共享用户登录信息，而不要求每个应用程序都单独进行用户登录。使用单点登录的应用程序对用户进行一次验证，然后此验证信息将被传播到其他所有涉及的应用程序。

单点登录适用于为同一区域和虚拟服务器配置的 Web 应用程序。

注：单点登录使用 HTTP Cookie 来传输令牌，该令牌将每个请求与已保存的用户标识关联起来，因此仅当浏览器客户机支持 Cookie 时才能使用单点登录。

单点登录根据以下规则运行：

- 当用户访问 Web 应用程序中受保护的资源时，服务器会要求用户使用为该 Web 应用程序定义的方法对自身进行验证。
- 经过验证之后， Application Server 将使用与该用户关联的角色在虚拟服务器中的所有 Web 应用程序之间进行授权决策，而不要求用户单独向每个应用程序进行验证。
- 当用户注销一个 Web 应用程序时（显式或由于会话失效），该用户在所有 Web 应用程序中的会话都将变为无效。此后，用户需要先登录才能访问任一应用程序中的受保护资源。

默认情况下，对 Application Server 的单点登录处于启用状态。要禁用单点登录或配置其他属性，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 展开要配置的实例：
   a. 要配置特定的实例，请展开该实例的配置节点。例如，对于默认实例 `server`，请展开 `server-config` 节点。
   b. 要为所有实例配置默认设置，请展开 `default-config` 节点。
3. 展开 “HTTP 服务”节点。
4. 展开 “虚拟服务器”节点，然后选择要配置为支持单点登录的虚拟服务器。
5. 单击 “添加属性”。

   一个空白的属性条目将被添加到列表末尾。

6. 在 “名称”字段中输入 sso-enable。
7. 在“值”字段中输入 false 禁用 SSO，输入 true 启用 SSO。默认情况下， SSO 处于启用状态。
8. 单击 “添加属性”并配置所有适用的 SSO 属性，来添加或更改任何其他单点登录属性。表 14-10 说明了有效的 SSO 属性。

表 14-10 虚拟服务器 SSO 属性

属性名称	说明	值
sso-max-inactive-seconds	如果未接收到任何客户机活动，在可以清除用户的单点登录记录之前所等待的时间（以秒为单位）。对虚拟服务器中的任何应用程序的访问都可以使单点登录记录保持活动状态。	默认值为 300 秒 （5 分钟）。值越高，为用户提供的持续性就越长，但服务器上的内存消耗也会越多。
sso-reap-interval-seconds	清除过期的单点登录记录的时间间隔 （以秒为单位）。	默认值为 60。

9. 单击 “保存”。
10. 如果控制台中显示 “需要重新启动”，请重新启动 Application Server。

有关连接器连接池的管理控制台任务

关于连接器连接池

连接器模块（也称为资源适配器）允许 J2EE 应用程序与企业信息系统 (EIS) 进行交互。**连接器资源**为应用程序提供了一个 EIS 连接。**连接器连接池**是一组用于特定 EIS 的可重复使用的连接。

安全映射允许在 J2EE 用户和组以及 EIS 用户和组之间创建映射。使用管理控制台可以创建、更新、列出和删除连接器连接池的安全映射。

注：在此上下文中，用户被称为主体。企业信息系统 (EIS) 是保存信息的任何系统。它可以是主机、消息传送系统、数据库系统或应用程序。

关于安全映射

使用安全映射可以将应用程序的调用者标识（主体或用户组）映射到容器管理的基于事务的方案中适当的 EIS 主体。当应用程序主体向 EIS 发出请求后，应用程序服务器将首先使用为连接器连接池定义的安全映射检查主体来确定已映射的后端 EIS 主体。如果没有完全匹配的主体，应用程序服务器将使用通配符字符规范（如果有）来确定已映射的后端 EIS 主体。当应用程序用户需要执行 EIS 操作（需要以 EIS 中的特定身份来执行）时，请使用安全映射。

创建安全映射

连接器连接池的安全映射将应用程序用户和组（主体）映射到 EIS 主体。当应用程序用户需要执行 EIS 操作（需要 EIS 中的特定身份）时，请使用安全映射。

要为给定的连接器连接池创建安全映射，请执行以下步骤。

1. 展开“资源”节点。
2. 展开“连接器”节点。
3. 选择“连接器连接池”节点。
4. 通过从当前池列表中选择连接器连接池的名称来选择一个连接器连接池，或通过从当前池列表中选择“新建”并按照“创建连接器连接池”中的说明创建新的连接器连接池。
5. 选择“安全映射”页面。
6. 单击“新建”以创建新的安全映射。

7. 在 “创建安全映射” 页面中，输入以下属性。
   - **名称**－输入要用于引用此特定安全映射的名称。
   - **用户组**－要映射到适当的 EIS 主体的应用程序的调用者身份。输入以逗号分隔的应用程序特定的用户组列表，或输入通配符星号 (*) 来表示所有用户或所有用户组。指定 “主体” 或 “用户组” 选项，但不能同时指定这两个选项。
   - **主体**－要映射到适当的 EIS 主体的应用程序的调用者身份。输入以逗号分隔的应用程序特定的主体列表，或输入通配符星号 (*) 来表示所有主体。指定 “主体” 或 “用户组” 选项，但不能同时指定这两个选项。
8. 在 “后端主体” 部分中，输入以下属性。
   - **用户名**－输入 EIS 用户名。企业信息系统 (EIS) 是保存信息的任何系统。它可以是主机、消息传送系统、数据库系统或应用程序。
   - **密码**－输入 EIS 用户的密码。
9. 单击 “确定” 以创建安全映射，或单击 “取消” 退出而不保存更改。

等效的 `asadmin` 命令为：`create-connector-security-map`

编辑安全映射

要修改给定的连接器连接池的安全映射，请执行以下步骤。

1. 展开 “资源” 节点。
2. 展开 “连接器” 节点。
3. 选择 “连接器连接池” 节点。
4. 通过从当前池列表中选择连接器连接池的名称来选择一个连接器连接池。
5. 选择 “安全映射” 页面。
6. 在 “安全映射” 页面中，从当前安全映射列表中选择一个安全映射。

7. 在 “编辑安全映射”页面中，根据需要修改以下属性。
   - **用户组** – 要映射到适当的 EIS 主体的应用程序的调用者身份。输入以逗号分隔的应用程序特定的用户组列表，或输入通配符星号 (*) 来表示所有用户或所有用户组。指定 “主体”或 “用户组”选项，但不能同时指定这两个选项。
   - **主体** – 要映射到适当的 EIS 主体的应用程序的调用者身份。输入以逗号分隔的应用程序特定的主体列表，或输入通配符星号 (*) 来表示所有主体。指定 “主体”或 “用户组”选项，但不能同时指定这两个选项。
8. 在 “后端主体”部分中，输入以下属性。
   - **用户名** – 输入 EIS 用户名。企业信息系统 (EIS) 是保存信息的任何系统。它可以是主机、消息传送系统、数据库系统或应用程序。
   - **密码** – 输入 EIS 用户的密码。
9. 单击 “保存”以保存对安全映射的更改。

有用的 asadmin 命令为：`list-connector-security-maps`, `update-connector-security-maps`

删除安全映射

要删除给定的连接器连接池的安全映射，请执行以下步骤。

1. 展开 “资源”节点。
2. 展开 “连接器”节点。
3. 选择 “连接器连接池”节点。
4. 通过从当前池列表中选择连接器连接池的名称来选择一个连接器连接池。
5. 选择 “安全映射”页面。
6. 在 “安全映射”页面中，单击要删除的安全映射的名称左侧的复选框。
7. 单击 “删除”。

等效的 asadmin 命令为：`delete-connector-security-map`

使用证书和 SSL

关于证书文件

安装 Application Server 时将生成一个适用于内部测试的 NSS 格式的数字证书。默认情况下， Application Server 将其证书信息存储到 *install_dir*/domains/*domain_name*/config 目录中的两个文件中：

- **密钥库文件** – 默认情况下，名为 key3.db，它包含 Application Server 的数字证书及其专用密钥。密钥库文件受密码保护。使用 asadmin change-master-password 命令可以更改该密码。有关 certutil 的更多信息，请参阅 “关于 CertUtil 实用程序”。

  每个密钥库条目都有唯一的别名。安装后， Application Server 密钥库会有一个别名为 slas 的条目。

- **信任存储文件** – 默认情况下，名为 cert8.db，它包含 Application Server 所信任的证书以及其他实体的公共密钥。对于信任的证书，服务器已确认证书中的公共密钥属于证书的拥有者。信任的证书通常包括那些证书授权机构 (CA) 的证书。

在 Platform Edition 的服务器端， Application Server 使用 JSSE 格式，该格式使用 keytool 来管理证书和密钥库。在 Enterprise Edition 的服务器端， Application Server 使用 NSS 格式，该格式使用 certutil 来管理存储专用密钥和证书的 NSS 数据库。两种版本的客户端 （应用程序客户端或独立客户端）上均使用 JSSE 格式。

默认情况下，使用密钥库和信任存储对 Application Server 进行配置，密钥库和信任存储将与示例应用程序配合使用并可用于开发。在用于生产目的时，您可能需要更改证书别名、将其他证书添加到信任存储或更改密钥库文件和信任存储文件的名称和/或位置。

更改证书文件的位置

供开发使用的密钥库和信任存储文件存储在 *install_dir*/domains/*domain_name*/config 目录中。要更改密钥库和信任存储文件的名称和/或位置，请执行以下步骤。

1. 在 “管理控制台”树中，展开 “配置”。
2. 展开 server-config （管理配置）节点。
3. 选择 “JVM 设置”节点。
4. 单击 “JVM 选项”选项卡。
5. 在 “JVM 选项”页面中，在 “值”字段中添加或修改以下值来反映证书文件的新位置：

   -Dcom.sun.appserv.nss.db=${com.sun.aas.instanceRoot}/*NSS_database_directory*

   其中， *ks_name* 是密钥库文件名， *ts_name* 是信任存储文件名。
6. 单击 “保存”。
7. 如果控制台中显示 “需要重新启动”，请重新启动 Application Server。

关于 Keytool 实用程序

在 Platform Edition 中使用 keytool 可以设置和使用 JSSE 数字证书。 J2SE SDK 附带了 keytool，因而允许管理员管理公共/专用密钥对和关联的证书。还允许用户高速缓存正与其通信的另一方的公共密钥 （以证书形式）。

要运行 keytool，必须先配置 shell 环境，以使 J2SE /bin 目录位于路径中，或者命令行中必须存在指向工具的完整路径。有关 keytool 的更多信息，请参见位于以下位置的 keytool 文档：

http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html

关于 CertUtil 实用程序

仅在 **Enterprise Edition** 中可以使用 `certutil` 来设置和使用 NSS 数字证书。证书数据库工具 `certutil` 是可以创建和修改 Netscape Communicator `cert8.db` 和 `key3.db` 数据库文件的命令行实用程序。该实用程序还可以列出、生成、修改或删除 `cert8.db` 文件中的证书，并可以创建或更改密码、生成新的公共和专用密钥对、显示密钥数据库的内容或删除 `key3.db` 文件中的密钥对。

密钥和证书管理进程通常以在密钥数据库中创建密钥开始，然后在证书数据库中生成和管理证书。下面列出的文档说明了如何使用 NSS 管理证书和密钥数据库，包括 `certutil` 实用程序的语法：

`http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html`

`pk12util` 是一个命令行实用程序，用于以 PKCS12 格式在证书 / 密钥数据库和文件之间导入和导出密钥及证书。有关 `pk12util` 实用程序的更多说明，请参阅：

`http://www.mozilla.org/projects/security/pki/nss/tools/pk12util.html`

有关使用 `certutil`、`pk12util` 和其他 NSS 安全性工具的更多信息，请参见位于 `http://www.mozilla.org/projects/security/pki/nss/tools` 的 NSS Security Tools。

这些工具位于 *install_dir*`/lib/` 目录。

生成服务器证书

使用 certutil 可以生成、导入和导出证书。有关如何生成、导入和导出证书的更多信息，请参阅 “关于 CertUtil 实用程序”。

为数字证书签名

创建数字证书之后，拥有者必须为其签名以防止伪造。电子商务站点或身份验证对其很重要的那些站点可以从知名的证书授权机构 (CA) 购买证书。如果无需考虑验证，例如当专用安全通信可以满足全部需求时，则可节省获取 CA 证书所花费的时间和费用并使用自签名证书。

使用来自 CA 的证书

要使用由 CA 签名的数字证书，请执行以下步骤：

1. 按照 CA Web 站点上的说明进行操作来生成证书密钥对。
2. 下载生成的证书密钥对。

   将证书保存在包含服务器密钥库和信任存储文件的目录中，默认情况下为 *install_dir*/domains/*domain-dir*/config 目录。有关更改此位置的说明，请参见 “更改证书文件的位置”。
3. 在 shell 中，切换到包含证书的目录。
4. 使用 certutil 将证书导入到本地密钥库和本地信任存储 （如有必要）。
5. 重新启动 Application Server。

有关使用 certutil 的完整信息，请参见位于以下位置的 certutil 文档：

http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html

删除证书

要删除现有证书，请使用 certutil 实用程序。有关 certutil 实用程序的更多信息，请参见 “关于 CertUtil 实用程序”。

详细信息

- 可以通过以下 URL 查看 Java 2 Standard Edition 的安全性讨论：
  http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html
- 可以通过以下 URL 查看 《 The J2EE 1.4 Tutorial 》的 "*Security*"：
  http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html
- **管理指南**的 “配置消息安全性”一章。
- Developer’s Guide 的 "*Securing Applications*" 一章。

第 15 章

配置消息安全性

本章介绍如何在 Sun Java System Application Server 8.1 2005Q1 中为 Web 服务配置消息层的安全性。本章包含以下主题：

- 关于消息安全性
- 有关消息安全性的管理控制台任务

本章中的某些内容假定您已对安全性和 Web 服务概念有了基本的了解。要详细了解这些概念，请在开始阅读本章之前先仔细阅读 “详细信息”中列出的内容。

关于消息安全性

- 消息安全性概述
- 了解 Application Server 中的消息安全性
- 确保 Web 服务的安全
- 确保样例应用程序的安全
- 配置 Application Server 以实现消息安全性

消息安全性概述

在**消息安全性**中，安全性信息插入到消息中以使其通过网络层传输，并和消息一起到达消息目标。消息安全性与传输层安全性不同 （在 《The J2EE 1.4 Tutorial》的 "*Security*" 一章中有说明），这是因为消息安全性可用于将消息保护从消息传输中分离开，从而使消息在传输后仍保持被保护状态。

Web **服务安全性**：*SOAP* **消息安全性** *(WS-Security)* 是可交互使用的 Web 服务安全性的国际标准，是在 OASIS 中由所有 Web 服务技术的主要提供商 （包括 Sun Microsystems）协作开发的。 WS-Security 是一种消息安全性机制，它使用 XML 加密和 XML 数字签名来确保 SOAP 上发送的 Web 服务消息的安全。 WS-Security 规范定义了多种安全令牌 （包括 X.509 证书、 SAML 断言和用户名 / 密码令牌）的用途，以验证和加密 SOAP Web 服务消息。

可以访问以下 URL 来查看 WS-Security 规范：

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

了解 Application Server 中的消息安全性

Sun Java System Application Server 8.1 2005Q1 将对 WS-Security 标准的支持集成至其 Web 服务客户机和服务器端容器。集成此功能，可以使 Web 服务安全性由代表应用程序的 Application Server 的容器强制执行，并且可以使此功能应用于保护任何 Web 服务应用程序而无需对应用程序的实现进行更改。 Application Server 通过提供工具将 SOAP 层消息安全性提供者和消息保护策略绑定到容器和容器中部署的应用程序，来达到此效果。

指定消息安全性职责

在 Sun Java System Application Server 8.1 2005Q1 中，系统管理员和应用程序部署者角色应主要负责配置消息安全性。尽管通常情况下，其他两个任一角色在无需更改应用程序实现的情况下就可以确保现有应用程序的安全而不必涉及开发者，但在某些情况下，应用程序开发者还是会有所作用的。以下小节定义了各种角色的职责：

- 系统管理员
- 应用程序部署者
- 应用程序开发者

系统管理员

系统管理员负责：

- 在 Application Server 中配置消息安全性提供者。
- 管理用户数据库。
- 管理密钥库和信任存储文件。
- 在使用加密并且运行 1.5.0 版之前的 Java SDK 时，配置 Java 加密扩展 (JCE) 提供者。
- 安装样例服务器。仅在 `xms` 样例应用程序用于演示消息层 Web 服务安全性的用途时，才执行此操作。

系统管理员使用管理控制台来管理服务器安全性设置，并使用命令行工具来管理证书数据库。在 PE 中，证书和专用密钥存储在密钥库中，并由 `keytool` 进行管理。SE 和 EE 将证书和专用密钥存储在 NSS 数据库中，并使用 `certutil` 对其进行管理。本文档主要针对系统管理员。有关消息安全性任务的概述，请参见 “配置 Application Server 以实现消息安全性”。

应用程序部署者

应用程序部署者负责：

- 在上游角色 （开发者或汇编者）尚未指定任何所需的特定于应用程序的消息保护策略时，指定这些策略 （在应用程序汇编时）。
- 修改特定于 Sun 的部署描述符以为 Web 服务端点和服务引用指定特定于应用程序的消息保护策略信息 （即 message-security-binding 元素）。

Developer's Guide 的 "*Securing Applications*" 一章中讨论了这些安全性任务。（有关指向本章的链接，请参见 “详细信息”。）

应用程序开发者

应用程序开发者可以打开消息安全性，但并不是必须要这样做。消息安全性可以由系统管理员设置，从而确保所有 Web 服务的安全；或者由应用程序部署者设置，这适用于绑定到应用程序的提供者或保护策略与绑定到容器的提供者或保护策略不同的情况。

应用程序开发者或汇编者负责：

- 确定应用程序是否需要特定于应用程序的消息保护策略。如果需要，则确保所需策略在应用程序汇编时指定，这可以通过与应用程序部署者沟通来完成。

关于安全令牌和安全机制

WS-Security 规范为使用安全令牌来验证和加密 SOAP Web 服务消息提供了可扩展机制。可以使用与 Application Server 一起安装的 SOAP 层消息安全性提供者来选取用户名 / 密码和 X509 证书安全令牌，以验证和加密 SOAP Web 服务消息。选取其他安全令牌 （包括 SAML 断言）的附加提供者将与 Application Server 的后续版本一起安装。

关于用户名令牌

Application Server 使用 SOAP 消息中的**用户名令**牌来建立消息**发件人**的验证标识。包含用户名令牌 （在嵌入式密码中）的消息的收件人通过确认发件人是否知道用户的秘密 （即密码），来验证消息发件人是否为经过授权的用户 （在令牌中标识）。

使用用户名令牌时，必须在 Application Server 中配置有效的用户数据库。有关该主题的更多信息，请参阅 “编辑区域”。

关于数字签名

Application Server 使用 XML 数字签名将验证标识绑定到消息**内容**中。客户机使用数字签名来建立它们的呼叫者标识，其方法与使用传输层安全性时用基本验证或 SSL 客户机证书验证建立呼叫者标识的方法相似。消息收件人将检验数字签名以验证消息内容的源 （可能与消息的发件人不同）。

使用数字签名时，必须在 Application Server 中配置有效的密钥库和信任存储文件。有关该主题的更多信息，请参阅 “关于证书文件”。

关于加密

加密的目的是将数据修改为只有目标读者能理解的形式。加密过程通过用加密元素代替原始内容来完成。像公共密钥加密指出的那样，可以使用加密来建立能够阅读消息的一方或多方的标识。

使用加密时，必须先安装支持加密的 JCE 提供者。有关该主题的更多信息，请参阅 “配置 JCE 提供者”。

关于消息保护策略

消息保护策略是针对消息处理请求和消息处理响应而定义的，并根据对源和 / 或收件人验证的要求来进行表达。源验证策略代表一个请求，即在消息中建立发送了消息或定义了消息内容的实体的标识，以使其可以由消息收件人进行验证。收件人验证策略代表一个请求，即发送消息，以使可以接收消息的实体的标识可以由消息发件人建立。提供者将应用特定的消息安全性机制以使消息保护策略在 SOAP Web 服务消息的上下文中实现。

在给容器配置提供者时，将定义请求和响应的消息保护策略。特定于应用程序的消息保护策略 （以 Web 服务端口或操作的粒度级别）也可以在应用程序或应用程序客户机的特定于 Sun 的部署描述符中进行配置。在任何情况下，如果定义了消息保护策略，则客户机请求和响应的消息保护策略必须与服务器请求和响应的消息保护策略相匹配 （即二者相当）。有关定义特定于应用程序的消息保护策略的更多信息，请参阅 Developer's Guide 的 "*Securing Applications*" 一章。“详细信息”中有指向该章的链接。

消息安全性术语表

以下介绍了此文档中所用到的术语。“配置 Application Server 以实现消息安全性”中也说明了这些概念。

- 验证层

  验证层是必须执行验证处理的消息层。Application Server 将在 SOAP 层强制执行 Web 服务消息安全性。

- 验证提供者

  在此版本的 Sun Java Systems Application Server 中， Application Server 调用**验证提供者**来处理 SOAP 消息层安全性。

  - **客户端提供者**可以建立 （通过签名或用户名/密码）请求消息的源标识和/或保护 （通过加密）请求消息，从而使这些消息只能由其目标收件人查看。客户端提供者还可以将其容器建立为收到的响应的已授权收件人 （通过成功地解密），并验证响应中的密码或签名来验证与此响应相关联的源标识。在 Application Server 中配置的客户端提供者可用于作为其他服务的客户机来保护服务器端组件 （即 Servlet 和 EJB）所发送的请求消息和所接收的响应消息。

  - **服务器端提供者**将其容器建立为收到的请求的已授权收件人 （通过成功地解密），并验证请求中的密码或签名以验证与该请求相关联的源标识。服务器端提供者还将建立 （通过签名或用户名/密码）响应消息的源标识和/或保护 （通过加密）响应消息，以使这些消息只能由其目标收件人查看。服务器端提供者仅由服务器端容器调用。

- 默认服务器提供者

  默认服务器提供者用于标识服务器提供者，系统将调用该服务器提供者以用于尚未绑定特定服务器提供者的任何应用程序。**默认服务器提供者**有时被称为**默认提供者**。

- 默认的客户机提供者

  默认客户机提供者用于标识客户机提供者，系统将调用该客户机提供者以用于尚未绑定特定客户机提供者的任何应用程序。

- 请求策略

  请求策略定义与验证提供者执行的请求处理关联的验证策略要求。按照消息发件人的顺序表达这些策略，从而使内容之后出现的加密请求表示消息收件人将在验证签名之前先要对消息进行解密。

- 响应策略

  响应策略定义与验证提供者执行的响应处理关联的验证策略要求。按照消息发件人的顺序表达这些策略，从而使内容之后出现的加密请求表示消息收件人将在验证签名之前先要对消息进行解密。

确保 Web 服务的安全

通过将 SOAP 层消息安全性提供者和消息保护策略绑定到部署有应用程序的容器或绑定到应用程序提供的 Web 服务端点，来确保在 Application Server 中部署的 Web 服务的安全。通过将 SOAP 层消息安全性提供者和消息保护策略绑定到客户机容器或绑定到由客户机应用程序声明的可移植服务引用，从而在 Application Server 的客户端容器中配置 SOAP 层消息安全性功能。

安装了 Application Server 后，将在 Application Server 的客户机和服务器端容器中配置 SOAP 层消息安全性提供者，这些提供者可由容器或容器中部署的各个应用程序或客户机进行绑定使用。在安装过程中，这些提供者将配置简单的消息保护策略，即如果被绑定到容器或者绑定到容器中的应用程序或客户机，将导致所有请求和响应消息中的内容源由 XML 数字签名进行验证。

可以采用 Application Server 的管理界面来绑定现有提供者以便供 Application Server 的服务器端容器使用，修改由提供者强制执行的消息保护策略，或使用替代的消息保护策略来创建新的提供者配置。“有关消息安全性的管理控制台任务”中定义了这些操作。可以在应用程序客户机容器的 SOAP 消息层安全性配置上执行类似的管理操作，如 “启用客户机应用程序的消息安全性”中所定义。

默认情况下， Application Server 中的消息层安全性处于禁用状态。要配置 Application Server 的消息层安全性，请按照 “配置 Application Server 以实现消息安全性”中列出的步骤进行操作。如果您要使 Web 服务安全性用于保护在 Application Server 上部署的所有 Web 服务应用程序，请按照 “启用消息安全性提供者”和 “启用客户机应用程序的消息安全性”中的步骤进行操作。

完成以上步骤 （可能包括重新启动 Application Server）后， Web 服务安全性将应用到 Application Server 中部署的所有 Web 服务应用程序。

配置特定于应用程序的 Web 服务安全性

通过在应用程序的特定于 Sun 的部署描述符中定义 message-security-binding 元素，可以配置特定于应用程序的 Web 服务安全性功能（在应用程序汇编时）。这些 message-security-binding 元素用于将特定提供者或消息保护策略与 Web 服务端点或服务引用相关联，并可以被限定以使这些元素应用到相应端点或引用服务的特定端口或方法。

有关定义特定于应用程序的消息保护策略的更多信息，请参阅 Developer's Guide 的 "*Securing Applications*" 一章。“详细信息”中有指向该章的链接。

确保样例应用程序的安全

Application Server 附带了名为 `xms` 的样例应用程序。`xms` 应用程序具有简单的 Web 服务功能，它由 J2EE EJB 端点和 Java Servlet 端点共同实现。这两个端点共享同一个服务端点接口。服务端点接口定义了单个操作 (`sayHello`)，此操作将获取一个字符串参数，并将由前置 `Hello` 编写的 `String` 返回到调用参数中。

将提供 `xms` 样例应用程序来演示 Application Server 的 WS-Security 功能的用途，以确保现有 Web 服务应用程序的安全。样例附带的说明介绍了如何启用 Application Server 的 WS-Security 功能以使其用于确保 `xms` 应用程序的安全。样例还将演示如何向应用程序直接绑定 WS-Security 功能（如“配置特定于应用程序的 Web 服务安全性”中所述）以使其特定应用于应用程序。

`xms` 样例应用程序安装在以下目录中：
install_dir`\samples\webservices\security\ejb\apps\xms\`

有关编译、封装和运行 `xms` 样例应用程序的信息，请参阅 Developer's Guide 的 "*Securing Applications*" 一章。“详细信息”中有指向该章的链接。

配置 Application Server 以实现消息安全性

Application Server 使用在其 SOAP 处理层中集成的消息安全性提供者来实现消息安全性。消息安全性提供者取决于 Application Server 的其他安全性工具。

要配置这些其他工具，请执行以下步骤：

1. 如果使用的 Java SDK 的版本早于 1.5.0，而且使用了加密技术，请配置 JCE 提供者。

   “配置 JCE 提供者”中说明了如何配置 JCE 提供者。

2. 如果使用了用户名令牌，请在必要时配置用户数据库。使用用户名/密码令牌时，必须配置适当的区域并为该区域配置适当的用户数据库。

   “编辑区域”中说明了如何配置用户数据库。

3. 管理证书和专用密钥 （如有必要）。

   “关于证书文件”中说明了如何管理证书和专用密钥。

如果已将 Application Server 的工具配置为供消息安全性提供者使用，则将会启用随 Application Server 一起安装的提供者，如 “启用消息安全性提供者”中所述。

配置 JCE 提供者

J2SE 1.4.x 中包括的 Java 加密扩展 (JCE) 提供者不支持 RSA 加密。由于由 WS-Security 所定义的 XML 加密通常是基于 RSA 加密，因此为了使用 WS-Security 来加密 SOAP 消息，您必须下载并安装支持 RSA 加密的 JCE 提供者。

注：RSA 是 RSA Data Security, Inc. 开发的公共密钥加密技术。 RSA 缩略词分别代表该技术的三位发明者：Rivest、 Shamir 和 Adelman。

如果是在 1.5 版的 Java SDK 中运行 Application Server，则已正确配置了 JCE 提供者。如果是在 1.4.x 版的 Java SDK 中运行 Application Server，请执行以下步骤以静态方式将 JCE 提供者添加为 JDK 环境的一部分。

1. 下载并安装 JCE 提供者 JAR （Java 归档）文件。通过以下 URL 可以获得支持 RSA 加密的 JCE 提供者的列表：

   `http://java.sun.com/products/jce/jce14_providers.html`

2. 将 JCE 提供者 JAR 文件复制到 <*JAVA_HOME*>`/jre/lib/ext/` 中。

3. 停止 Application Server。如果未停止 Application Server 而后来又在该进程中重新启动了 Application Server，则 Application Server 将无法识别 JCE 提供者。

4. 在任何一种文本编辑器中编辑 *<JAVA_HOME>*/jre/lib/security/java.security 属性文件。将刚才下载的 JCE 提供者添加到此文件。 java.security 文件包含添加该提供者的详细说明。通常，您需要在具有类似属性的某个位置处添加一行，其格式如下：

   ```
   security.provider.<n>=<provider class name>
   ```

   在本示例中， *<n>* 是 Application Server 评估安全性提供者时使用的优先级顺序。将刚才添加的 JCE 提供者的 *<n>* 值设置为 2。

   例如，如果下载的是 Legion of the Bouncy Castle JCE 提供者，则应添加以下行。

   ```
   security.provider.2=org.bouncycastle.jce.provider.
   BouncyCastleProvider
   ```

   确保将 Sun 安全性提供者保持在最高优先级，值为 1。

   ```
   security.provider.1=sun.security.provider.Sun
   ```

   将其他安全性提供者的优先级调低，从而使每个优先级上只有一个安全性提供者。

   以下是提供所需 JCE 提供者并将现有提供者保持在正确位置的 java.security 文件的示例。

   ```
   security.provider.1=sun.security.provider.Sun
   security.provider.2=org.bouncycastle.jce.provider.
   BouncyCastleProvider
   security.provider.3=com.sun.net.ssl.internal.ssl.Provider
   security.provider.4=com.sun.rsajca.Provider
   security.provider.5=com.sun.crypto.provider.SunJCE
   security.provider.6=sun.security.jgss.SunProvider
   ```

5. 保存并关闭文件。

6. 重新启动 Application Server。

有关消息安全性的管理控制台任务

为使用消息安全性而对 Application Server 进行设置的大部分步骤都可以通过使用管理控制台、 asadmin 命令行工具或通过手动编辑系统文件来完成。通常，不建议使用编辑系统文件，因为它可能会导致出现无意识的更改而使 Application Server 不能正常运行，所以，如有可能，建议优先选择使用管理控制台来配置 Application Server，其次选择使用 asadmin 工具命令。仅在无管理控制台或等效的 asadmin 时，才使用手动编辑系统文件步骤。

以 （可插入）验证模块的形式将消息层安全性支持集成到 Application Server 及其客户机容器中。默认情况下， Application Server 中的消息层安全性处于禁用状态。以下各节提供了启用、创建、编辑和删除消息安全性配置和提供者的详细信息。

- 启用消息安全性提供者
- 配置消息安全性提供者
- 创建消息安全性提供者
- 删除消息安全性配置
- 删除消息安全性提供者
- 启用客户机应用程序的消息安全性

大多数情况下，在执行以上列出的管理操作后应重新启动 Application Server。它尤其适用于当执行完操作时，您要将管理更改的效果应用到 Application Server 上已部署的应用程序中的情况。

启用消息安全性提供者

要为在 Application Server 中部署的 Web 服务端点启用消息安全性，必须指定要在服务器端默认使用的提供者。如果启用了消息安全性的默认提供者，您还需要启用要由 Application Server 中部署的 Web 服务客户机所使用的提供者。“启用客户机应用程序的消息安全性”中说明了有关启用客户机使用的提供者的信息。

要为源于已部署的端点的 Web 服务调用启用消息安全性，您必须指定默认客户机提供者。如果已为 Application Server 启用了默认客户机提供者，则必须确保从 Application Server 中部署的端点中调用的所有服务均已配置为与消息层安全性兼容。

要为 Application Server 启用默认提供者，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 展开 “安全性”节点。
4. 展开 “消息安全性”节点。
5. 选择 "SOAP" 节点。
6. 选择 “消息安全性”选项卡。
7. 在 “编辑消息安全性配置”页面中，指定一个要用于服务器端的提供者和一个要用于客户端的提供者，以用于尚未绑定特定提供者的所有应用程序。通过修改以下可选属性来完成此操作：
   - **默认提供者**－要为尚未绑定特定服务器提供者的任何应用程序调用的服务器提供者的标识。

     默认情况下，没有为 Application Server 选择任何提供者配置。要标识服务器端提供者，请选择 `ServerProvider`。选择空选项意味着不会在服务器端（默认情况下）调用消息安全性提供者。

     通常应在该字段中选择 `ServerProvider`。
   - **默认客户机提供者**－要为尚未绑定特定客户机提供者的任何应用程序调用的客户机提供者的标识。

     默认情况下，没有为 Application Server 选择任何提供者配置。要标识客户端提供者，请选择 `ClientProvider`。选择空选项意味着不会在客户端 （默认情况下）调用消息安全性提供者。

     通常应在该字段中选择空。如果您要启用默认提供者和消息保护策略以应用到源于 Application Server 上所部署的 Web 服务端点的 Web 服务调用，则应选择 ClientProvider。
8. 单击 “保存”。
9. 如果您已启用客户机或服务器提供者并且要修改已启用的提供者的消息保护策略，请参阅 “配置消息安全性提供者”以获得有关修改在此步骤中所启用的消息安全性提供者的配置的信息。

等效的 asadmin 命令为：

- 要指定默认的服务器提供者，请使用：

  ```
  asadmin set --user <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  default_provider=ServerProvider
  ```

- 要指定默认的客户机提供者，请使用：

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  default_client_provider=ClientProvider
  ```

配置消息安全性提供者

通常，应重新配置提供者以修改其消息保护策略，当然提供者类型、实现类和特定于提供者的配置属性可能也需要修改。请按以下所列步骤重新配置消息安全性提供者。

1. 在管理控制台的树组件中，展开 “配置” 节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 展开 “安全性” 节点。
4. 展开 “消息安全性” 节点。
5. 选择 "SOAP" 节点。
6. 选择 “提供者” 选项卡。
7. 选择要编辑的消息安全性提供者。Application Server 附带了 `ClientProvider` 和 `ServerProvider`。

8. 在 “编辑提供者配置”页面的 “提供者配置”部分，可以修改以下属性：

   - **提供者类型**－选择 client、 server 或 client-server 以确定是将提供者用作客户机验证提供者、服务器验证提供者还是兼用作两者 （客户机 - 服务器提供者）。
   - **类名**－输入提供者的 Java 实现类。客户机验证提供者必须实现 com.sun.xml.wss.provider.ClientSecurityAuthModule 接口。服务器端提供者必须实现 com.sun.xml.wss.provider.ServerSecurityAuthModule 接口。一个提供者可以实现这两种接口，但该提供者必须实现对应于其提供者类型的接口。

9. 在 “创建提供者配置”页面的 “请求策略”部分，根据需要输入以下**可选**值。这些属性是可选的，但如果不指定这些属性，将不会对请求消息执行任何验证。

   请求策略定义与验证提供者执行的请求处理关联的验证策略要求。按照消息发件人的顺序表达这些策略，从而使内容之后出现的加密请求表示消息收件人将在验证签名之前先要对消息进行解密。

   - **验证源**－选择 sender、 content 或空 （空白选项）来定义对消息层发件人验证的要求 （例如用户名密码）、内容验证 （例如数字签名）或不对请求消息执行任何验证。如果指定为空，则不需要进行请求的源验证。
   - **验证收件人**－选择 beforeContent 或 afterContent 来定义向发件人响应消息的收件人的消息层验证的请求 （例如，通过 XML 加密）。如果不指定该值，则其为默认值 afterContent。

   有关为产生以下消息保护策略而由 SOAP 消息安全性提供者执行的操作的说明，请参见 “请求策略配置和响应策略配置的操作”。

10. 在 “创建提供者配置”页面的 “响应策略”部分，根据需要输入以下**可选**属性。这些属性是可选的，但如果不指定这些属性，将不会对响应消息执行任何验证。

    响应策略定义与验证提供者执行的响应处理关联的验证策略要求。按照消息发件人的顺序表达这些策略，从而使内容之后出现的加密请求表示消息收件人将在验证签名之前先要对消息进行解密。

    - **验证源**－选择 sender、 content 或空 （空白选项）来定义要应用到响应消息的消息层发件人验证的请求 （例如用户名口令）或内容验证 （例如数字签名）。如果指定为空，则不需要进行响应的源验证。
    - **验证收件人**－选择 beforeContent 或 afterContent 来定义向发件人响应消息的收件人的消息层验证的请求 （例如，通过 XML 加密）。如果不指定该值，则其为默认值 afterContent。

    有关为产生以下消息保护策略而由 SOAP 消息安全性提供者执行的操作的说明，请参见 “请求策略配置和响应策略配置的操作”。

11. 通过单击 “添加属性”按钮添加其他属性。 Application Server 附带的提供者支持下面列出的属性。如果使用了其他提供者，则有关属性和有效值的更多信息，请参阅其他提供者的文档。

    - server.config 一包含服务器配置信息的 XML 文件的目录和文件名。例如，该值为 *install_dir*/domains/*domain_dir*/config/wss-server-config.xml。

12. 单击 “保存”。

下面列出了等效的 asadmin 命令。要设置响应策略，请用 response 一词替换以下命令中的 request 一词。

- 要将请求策略添加到客户机并设置验证源，请使用：

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  provider-config.ClientProvider.request-policy.auth_source=
  <sender | content>
  ```

- 要将请求策略添加到服务器并设置验证源，请使用：

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  provider-config.ServerProvider.request-policy.auth_source=
  <sender | content>
  ```

- 要将请求策略添加到客户机并设置验证收件人，请使用：

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  provider-config.ClientProvider.request-policy.auth_recipient=
  <before-content | after-content>
  ```

- 要将请求策略添加到服务器并设置验证收件人，请使用：

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  provider-config.ServerProvider.request-policy.auth_recipient=
  <before-content | after-content>
  ```

创建消息安全性提供者

要创建新的消息安全性提供者，请执行以下步骤。要配置现有提供者，请执行 “配置消息安全性提供者”中的步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 展开 “安全性”节点。
4. 展开 “消息安全性”节点。
5. 选择 "SOAP" 节点。
6. 选择 “提供者”选项卡。
7. 在 “提供者配置”页面中，单击 “新建”。
8. 在 “创建提供者配置”页面的 “提供者配置”部分，输入以下信息：
   - **默认提供者**－选中该字段旁边的框，以使新消息安全性提供者成为要为尚未绑定特定提供者的任何应用程序调用的提供者。提供者是成为默认客户机提供者、默认服务器提供者还是两者兼而有之，取决于为 “提供者类型”所选的值。
   - **提供者类型**－选择 `client`、 `server` 或 `client-server` 以确定是将提供者用作客户机验证提供者、服务器验证提供者还是兼用作两者 （客户机 - 服务器提供者）。
   - **提供者 ID** －输入该提供者配置的标识符。此名称将显示在 “当前提供者配置”列表中。
   - **类名**－输入提供者的 Java 实现类。客户机验证提供者必须实现 `com.sun.xml.wss.provider.ClientSecurityAuthModule` 接口。服务器端提供者必须实现 `com.sun.xml.wss.provider.ServerSecurityAuthModule` 接口。一个提供者可以实现这两种接口，但该提供者必须实现对应于其提供者类型的接口。

9. 在 “创建提供者配置”页面的 “请求策略”部分，根据需要输入以下可选值。这些属性是**可选**的，但如果不指定这些属性，将不会对请求消息执行任何验证。

   - **验证源**－选择 `sender`、 `content` 或空 （空白选项）来定义对消息层发件人验证的要求 （例如用户名密码）、内容验证 （例如数字签名）或不对请求消息执行任何验证。如果指定为空，则不需要进行请求的源验证。
   - **验证收件人**－选择 `beforeContent` 或 `afterContent` 来定义向发件人响应消息的收件人的消息层验证的请求 （例如，通过 XML 加密）。如果不指定该值，则其为默认值 `afterContent`。

   有关为产生以下消息保护策略而由 SOAP 消息安全性提供者执行的操作的说明，请参见 “请求策略配置和响应策略配置的操作”。

10. 在 “创建提供者配置”页面的 “响应策略”部分，根据需要输入以下**可选**属性。这些属性是可选的，但如果不指定这些属性，将不会对响应消息执行任何验证。

    - **验证源**－选择 `sender`、 `content` 或空 （空白选项）来定义要应用到响应消息的消息层发件人验证的请求 （例如用户名口令）或内容验证 （例如数字签名）。如果指定为空，则不需要进行响应的源验证。
    - **验证收件人**－选择 `beforeContent` 或 `afterContent` 来定义向发件人响应消息的收件人的消息层验证的请求 （例如，通过 XML 加密）。如果不指定该值，则其为默认值 `afterContent`。

    有关为产生以下消息保护策略而由 SOAP 消息安全性提供者执行的操作的说明，请参见 “请求策略配置和响应策略配置的操作”。

11. 通过单击 “添加属性”按钮添加其他属性。 Application Server 附带的提供者支持下面列出的属性。如果使用了其他提供者，则有关属性和有效值的更多信息，请参阅其他提供者的文档。

    - `server.config` －包含服务器配置信息的 XML 文件的目录和文件名。例如，该值为 *install_dir*`/domains/`*domain_dir*`/config/wss-server-config.xml`。

12. 单击 “确定”保存此配置，或单击 “取消”退出而不保存更改。

等效的 `asadmin` 命令为：`create-message-security-provider`

请求策略配置和响应策略配置的操作

表 15-1 显示了消息保护策略配置以及由该配置的 WS-Security SOAP 消息安全性提供者所执行的最终消息安全性操作。

表 15-1 消息保护策略与 WS-Security SOAP 消息安全性操作的映射

消息保护策略	最终的 WS-Security SOAP 消息保护操作
auth-source="sender"	此消息包含 wsse:Security 标头，此标头包含 wsse:UsernameToken（带有密码）。
auth-source="content"	对 SOAP 消息主体的内容进行签名。此消息包含 wsse:Security 标头，此标头包含显示为 ds:Signature 的消息主体签名。
auth-source="sender" auth-recipient="before-content" 或 auth-recipient="after-content"	SOAP 消息主体的内容已加密并用最终的 xend:EncryptedData 替换。此消息包含 wsse:Security 标头，此标头包含 wsse:UsernameToken（带有密码）和 xenc:EncryptedKey。xenc:EncryptedKey 包含用于加密 SOAP 消息主体的密钥。此密钥在收件人的公共密钥中加密。
auth-source="content" auth-recipient="before-content"	SOAP 消息主体的内容已加密并用最终的 xend:EncryptedData 替换。对 xenc:EncryptedData 进行签名。此消息包含 a wsse:Security 标头，此标头包含 xenc:EncryptedKey 和 ds:Signature。 xenc:EncryptedKey 包含用于加密 SOAP 消息主体的密钥。此密钥在收件人的公共密钥中加密。
auth-source="content" auth-recipient="after-content"	对 SOAP 消息主体的内容进行签名、加密，并用最终的 xend:EncryptedData 替换。此消息包含 wsse:Security 标头，此标头包含 xenc:EncryptedKey 和 dsSignature。 xenc:EncryptedKey 包含用于加密 SOAP 消息主体的密钥。此密钥在收件人的公共密钥中加密。
auth-recipient="before-content" 或 auth-recipient="after-content"	SOAP 消息主体的内容已加密并用最终的 xend:EncryptedData 替换。此消息包含 wsse:Security 标头，此标头包含 xenc:EncryptedKey。 xenc:EncryptedKey 包含用于加密 SOAP 消息主体的密钥。此密钥在收件人的公共密钥中加密。
未指定策略。	模块未执行任何安全操作。

删除消息安全性配置

要删除消息安全性配置，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 展开 “安全性”节点。
4. 选择 “消息安全性”节点。
5. 单击要删除的消息安全性配置左侧的复选框。
6. 单击 “删除”。

删除消息安全性提供者

要删除消息安全性提供者，请执行以下步骤。

1. 在管理控制台的树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 展开 “安全性”节点。
4. 展开 “消息安全性”节点。
5. 选择 "SOAP" 节点。
6. 选择 “提供者”页面。
7. 单击要删除的提供者配置左侧的复选框。
8. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-message-security-provider`

启用客户机应用程序的消息安全性

必须配置客户机提供者的消息保护策略，以使其等效于将与其进行交互的服务器端提供者的消息保护策略。在安装 Application Server 时已配置（但未启用）的提供者已符合此情况。

要启用客户机应用程序的消息安全性，请修改应用程序客户机容器特定于 Sun Java System Application Server 的配置。

要启用应用程序客户机中的默认客户机提供者，请执行以下步骤：

1. 停止所有取决于客户机容器描述符的客户机应用程序。
2. 在文本编辑器中打开 Sun 应用程序客户机容器描述符，该描述符位于 *install_dir*/domains/*domain_dir*/config/sun-acc.xml。
3. 将粗体文本添加到文件中以启用应用程序客户机中的默认客户机提供者。还提供了其他代码，以显示启用客户机应用程序的消息安全性的代码所在的位置。非粗体代码在安装中可能稍有不同。请勿更改这些非粗体文本。

```
<client-container>
  <target-server name="<your_host>" address="<your_host>" port="<your_port>"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
    default-client-provider="ClientProvider">
    <provider-config
      class-name="com.sun.xml.wss.provider.ClientSecurityAuthModule"
      provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender"/>
      <response-policy/>
       <property name="security.config"
      value="C:/Sun/AppServer/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

客户机容器中配置的消息安全性提供者还需要具有对专用密钥和信任证书的访问权限。这将通过为以下系统属性（位于应用程序客户机启动脚本中）定义适当的值来完成。

-Djavax.net.ssl.keyStore

-Djavax.net.ssl.trustStore

设置应用程序客户机配置的请求策略和响应策略

请求策略和响应策略定义与验证提供者执行的请求处理和响应处理关联的验证策略要求。按照消息发件人的顺序表达这些策略，从而使内容之后出现的加密请求表示消息收件人将在验证签名之前先要对消息进行解密。

要获得消息安全性，必须既在服务器中也在客户机中启用请求策略和响应策略。在客户机和服务器中配置策略时，请确保请求 / 响应应用程序级别消息绑定的保护的客户机策略与服务器策略匹配。

要设置应用程序客户机配置的请求策略，请按照“启用客户机应用程序的消息安全性”中所述修改应用程序客户机容器的 Sun Java System Application Server 特定的配置。在应用程序客户机配置文件中，添加**粗体**文本以设置请求策略。提供的其他代码可用作参考。非粗体代码在安装中可能稍有不同。请勿更改这些非粗体文本。

```
<client-container>
  <target-server name="<your_host>" address="<your_host>" port="<your_port>"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
    default-client-provider="ClientProvider">
    <provider-config
      class-name="com.sun.xml.wss.provider.ClientSecurityAuthModule"
      provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
       <property name="security.config"
      value="install_dir/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

auth-source 的有效值包括 sender 和 content。 auth-recipient 的有效值包括 before-content 和 after-content。“请求策略配置和响应策略配置的操作”中包含一个表，用于说明这些值的各种组合的结果。

如果不想指定请求策略或响应策略，请将该元素保留为空，例如：

```
<response-policy/>
```

详细信息

- 可以通过以下 URL 查看 Java 2 Standard Edition 的安全性讨论：
  http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html
- 可以从以下 URL 查看 《The J2EE 1.4 Tutorial》的 "*Security*" 一章：
  http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html
- **管理指南**的 “配置安全性”一章。
- Developer’s Guide 的 "*Securing Applications*" 一章。
- 可以从以下 URL 查看 《Oasis Web Services Security: SOAP Message Security (WS-Security)》规范：
  http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
- 可以从以下 URL 中找到 《OASIS Web Services Security Username Token Profile 1.0》：
  http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
- 可以从以下 URL 中找到 《OASIS Web Services Security X.509 Certificate Token Profile 1.0》：
  http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
- 可以通过以下 URL 查看 XML-Signature Syntax and Processing 文档：
  http://www.w3.org/TR/xmldsig-core/
- 可以通过以下 URL 查看 XML Encryption Syntax and Processing 文档：
  http://www.w3.org/TR/xmlenc-core/

第 16 章

事务

通过将一个或多个步骤装入不可分的工作单元，事务可确保数据的完整性和一致性。本章包括以下几个部分：

关于事务

什么叫事务？

事务是应用程序中一系列严密的操作，所有操作必须成功完成，否则在每个操作中所作的所有更改都会被撤消。例如，将资金从支票帐户转到储蓄帐户中是一项事务，按步骤如下进行：

1. 检查支票帐户是否有足够的资金来支付此转帐操作。
2. 如果支票帐户中有足够的资金，则将该笔资金记入此帐户的借方。
3. 将这些资金记入储蓄帐户的贷方。
4. 将此次转帐记录到支票帐户日志中。
5. 将此次转帐记录到储蓄帐户日志中。

如果这些步骤的任何一个步骤失败，则必须撤消在前面的步骤中所做的所有更改，而且支票帐户和储蓄帐户的状态必须与它们在事务开始之前的状态相同。该事件称为**回滚**。如果所有步骤都成功完成，则该事务处于**提交**状态。事务以提交或回滚结束。

J2EE 技术中的事务

J2EE 技术中的事务处理包括以下五个参与者：

- 事务管理器
- Application Server
- 资源管理器
- 资源适配器
- 用户应用程序

通过实现各自的 API 和功能，每个实体均有助于提高事务处理的可靠性，如下所述：

- 事务管理器提供了支持事务划分、事务资源管理、同步和事务上下文传播所需的服务和管理功能。
- Application Server 提供了支持应用程序运行时环境 （包含事务状态管理）所需的基础结构。
- 应用程序可以使用资源管理器 （通过资源适配器）访问资源。资源管理器通过实现事务管理器所使用的事务资源接口来参与分布式事务，从而就事务关联、事务完成及恢复工作等过程进行通信。例如，关系型数据库服务器就是这样的资源管理器。
- 资源适配器是一个系统级的软件库，应用程序服务器或客户机可使用它连接到资源管理器。资源适配器通常专用于资源管理器。它以库的形式存在，并在使用它的客户机地址空间中使用。例如， JDBC 驱动程序就是这样的资源适配器。
- 为在应用程序服务器环境中运行而开发的事务用户应用程序可以使用 JNDI 查找事务数据源及事务管理器 （可选）。应用程序可以使用企业 Bean 的声明事务属性设置或明确的程序事务分界。

有关事务的管理控制台任务

Application Server 将基于管理控制台中的设置来处理事务。

配置事务

本节说明了配置以下事务属性的过程：

- 事务恢复
- 事务超时
- 事务日志记录

事务恢复

由于服务器崩溃或资源管理器崩溃，事务可能未完成。完成这些出现问题的事务并从故障中恢复是至关重要的。Application Server 可在服务器启动时从这些故障中恢复并完成事务。

执行恢复操作时，如果无法访问某些资源，则服务器重新启动操作可能被延迟，因为服务器正在尝试恢复事务。

如果事务跨服务器进行，启动该事务的服务器会联系其他服务器以获得事务的结果。如果无法访问其他服务器，则该事务将使用 “试探性决定”字段来确定结果。

要配置 Application Server 从事务中恢复的方法，请执行以下步骤：

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 选择 “事务服务”节点。
4. 要启用恢复未完成事务的操作，请在 “重新启动时”字段中选取 “恢复”。
5. 在 “重试超时”字段中，设置 Application Server 尝试连接无法访问的服务器的时间值 （以秒为单位）。默认值为 10 分钟 （600 秒）。

6. 在 “试探性决定”字段中，为事务中无法访问的服务器设置策略。

   除非有充分的理由将此字段设置为 “提交”，否则请将 “试探性决定”保留设置为 “回滚”。提交不确定的事务会破坏应用程序的数据完整性。

7. 单击 “保存”。
8. 重新启动 Application Server。

事务超时

默认情况下，服务器不会使事务超时。即，服务器无限期地等待事务完成。如果为事务设置了超时值，而事务在配置的时间内未完成，则 Application Server 将回滚此事务。

要设置超时值，请执行以下步骤：

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 选择 “事务服务”节点。
4. 在 “事务超时”字段中，输入事务超时之前等待的秒数。

   事务超时的默认值为 0 秒。此值禁用事务超时。

5. 单击 “保存”。
6. 重新启动 Application Server。

事务日志记录

为了保持被调用资源的数据完整性，同时为了能够从故障中恢复，事务日志将记录有关每个事务的信息。事务日志保存在 “事务日志位置”字段指定的目录的 `tx` 子目录中。用户无法读取这些日志。

要设置事务日志的位置，请执行以下步骤：

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要配置所有实例的默认设置，请选择 default-config 节点。
3. 选择 “事务服务”节点。
4. 在 “事务日志位置”字段中输入事务日志的位置。

   将创建 tx 子目录，而且事务日志将被保存在该目录下。

   默认值为 ${com.sun.aas.instanceRoot}/logs。
   ${com.sun.aas.instanceRoot} 变量是实例的名称，并在您启动一个 Application Server 实例时设置此变量。要查看 ${com.sun.aas.instanceRoot} 的值，请单击 “实际值”。
5. 单击 “保存”。
6. 重新启动 Application Server。

密钥点操作可以压缩事务日志文件。密钥点间隔是日志中密钥点操作之间的事务数量。密钥点操作可以减小事务日志文件的大小。密钥点间隔数越大 （例如，2048），事务日志文件也越大，但密钥点操作较少，性能可能更佳。密钥点间隔越小 （例如， 256），日志文件也越小，而同时由于密钥点操作较为频繁，性能会略微降低。

要设置密钥点间隔，请执行以下步骤：

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要配置所有实例的默认设置，请选择 default-config 节点。
3. 选择 “事务服务”节点。
4. 在 “密钥点间隔”字段中，输入密钥点操作之间的事务数量。

   默认值为 2048。
5. 单击 “保存”。
6. 重新启动 Application Server。

第 17 章

配置 HTTP 服务

本章介绍了如何为 Application Server 的 HTTP 服务组件配置虚拟服务器和 HTTP 侦听器。

关于 HTTP 服务

什么是 HTTP 服务？

HTTP 服务是 Application Server 的一个组件，它提供了一些工具来部署 Web 应用程序，并使 HTTP 客户机可以访问已部署的 Web 应用程序。（请参见第 111 页上的“部署 Web 应用程序”。）这些工具是借助于以下两种相关对象提供的：虚拟服务器和 HTTP 侦听器。

虚拟服务器

虚拟服务器 （有时也称为虚拟主机）是一个允许同一个物理服务器来托管多个 Internet 域名的对象。同一个物理服务器上托管的所有虚拟服务器共享该物理服务器的 Internet 协议 (IP) 地址。虚拟服务器将某个服务器的域名 （例如 `www.aaa.com`）与运行 Application Server 的特定服务器关联起来。

注：请勿将 Internet 域与 Application Server 的管理域混淆。

例如，假设您希望在物理服务器上托管以下这些域：

```
www.aaa.com
www.bbb.com
www.ccc.com
```

同时假设 `www.aaa.com`.`www.bbb.com` 和 `www.ccc.com` 都分别具有与之关联的 Web 模块 `web1`.`web2` 和 `web3`。

这意味着以下 URL 将全部由您的物理服务器处理：

```
http://www.aaa.com:8080/web1
http://www.bbb.com:8080/web2
http://www.ccc.com:8080/web3
```

第一个 URL 将被映射到虚拟主机 `www.aaa.com`，第二个 URL 将被映射到虚拟主机 `www.bbb.com`，第三个 URL 将被映射到虚拟主机 `www.ccc.com`。

另一方面，由于未向 `www.bbb.com` 注册 `web3`，以下 URL 将导致 404 返回码：

```
http://www.bbb.com:8080/web3
```

要使此映射有效，请确保 `www.aaa.com`.`www.bbb.com` 和 `www.ccc.com` 均可解析为物理服务器的 IP 地址。需要向您的网络的 DNS 服务器注册这些域名。此外，在 UNIX 系统上，应将这些域添加到 `/etc/hosts` 文件中 （如果 `/etc/nsswitch.conf` 文件中的 `hosts` 设置包括 `files`）。

启动 Application Server 时，将自动启动以下虚拟服务器：

- 名为 `server` 的虚拟服务器，用于托管所有用户定义的 Web 模块
- 名为 `asadmin` 的虚拟服务器，用于托管所有与管理相关的 Web 模块 （特别是管理控制台）。该服务器是一个受限制的服务器，您不能将 Web 模块部署到该虚拟服务器上。

如果是在非生产环境中开发、测试和部署 Web 服务，通常您只需要使用 `server` 虚拟服务器。在生产环境中，其他虚拟服务器可以为用户和客户提供托管工具，这样，尽管只有一个物理服务器，但每个用户和客户好像都有自己的 Web 服务器。

HTTP 侦听器

每个虚拟服务器都通过一个或多个 HTTP 侦听器来提供服务器与客户机之间的连接。每个 HTTP 侦听器都是包含 IP 地址、端口号、服务器名和默认虚拟服务器的侦听套接字。

HTTP 侦听器必须有唯一的端口号和 IP 地址的组合。例如，通过将 IP 地址指定为 0.0.0.0， HTTP 侦听器可以在计算机的给定端口上侦听所有的已配置 IP 地址。或者， HTTP 侦听器可以为每个侦听器指定唯一的 IP 地址，但使用相同的端口。

由于 HTTP 侦听器是 IP 地址和端口号的组合，因此您可以拥有多个 IP 地址相同但端口号不同 （如 1.1.1.1:8081 和 1.1.1.1:8082）的 HTTP 侦听器，或 IP 地址不同但端口号相同 （如 1.1.1.1:8081 和 1.2.3.4:8081）的 HTTP 侦听器 （如果已将计算机配置为可以响应这些地址）。

但是，如果 HTTP 侦听器使用 0.0.0.0 IP 地址侦听某个端口上的所有 IP 地址，则不能使用其他 IP 地址创建用于侦听同一端口上的特定 IP 地址的 HTTP 侦听器。例如，如果 HTTP 侦听器使用 0.0.0.0:8080 （端口 8080 上的所有 IP 地址），则其他 HTTP 侦听器不能使用 1.2.3.4:8080。

由于运行 Application Server 的系统通常只能访问一个 IP 地址，因此 HTTP 侦听器通常使用 0.0.0.0 IP 地址和不同的端口号，其中每个端口号用于不同目的。如果系统可以访问多个 IP 地址，则每个地址可以用于不同目的。

默认情况下， Application Server 启动时，它具有以下 HTTP 侦听器：

- 两个分别名为 `http-listener-1` 和 `http-listener-2` 的 HTTP 侦听器，这两个侦听器与名为 `server` 的虚拟服务器相关联。名为 `http-listener-1` 的侦听器未启用安全性；而名为 `http-listener-2` 的侦听器启用了安全性。
- 名为 `admin-listener` 的 HTTP 侦听器，该侦听器与名为 `asadmin` 的虚拟服务器相关联。该侦听器已启用安全性。

所有这些侦听器均使用 IP 地址 0.0.0.0 和在安装 Application Server 过程中被指定为 HTTP 服务器端口号的端口号。如果 Application Server 使用默认端口号值，则 `http-listener-1` 使用端口 8080， `http-listener-2` 使用端口 8181， `admin-listener` 使用端口 4849。

每个 HTTP 侦听器均有一个默认虚拟服务器。当请求 URL 的主机部分与 HTTP 侦听器关联的任何虚拟服务器 （在虚拟服务器的 `http-listeners` 属性中列出 HTTP 侦听器，即可将虚拟服务器与该 HTTP 侦听器关联起来）均不匹配时， HTTP 侦听器会将请求 URL 路由到默认虚拟服务器。

此外，还应在 HTTP 侦听器中指定接收器线程的数目。接收器线程就是等待连接的线程。它们用于接收连接并将其置于队列 （称为连接队列）中，以便随后由工作线程拾取连接。您需要配置足够多的接收器线程，以便在新请求传入时始终有一个可用的线程，但是，线程数目不能过多，否则会占用过多的系统资源。连接队列中既包括接收器线程刚刚接收的新连接又包括由保持活动连接管理子系统管理的持久性连接。

一组请求处理线程将从连接队列中检索传入的 HTTP 请求并处理这些请求。这些线程将解析 HTTP 头、选择相应的虚拟服务器并通过请求处理引擎以处理请求。如果没有更多要处理的请求，但连接可以保持持久性 （通过使用 HTTP/1.1 或发送 `Connection:keep-alive` 标头），请求处理线程将假定连接处于空闲状态，并将连接传递给保持活动连接管理子系统。

保持活动子系统将定期轮询此类空闲连接，并在连接队列中对那些活动的连接进行排队，以便将来进行处理。请求处理线程将再次从连接队列中检索连接并处理其请求。保持活动子系统是多线程的，可以管理大约数万个连接。通过将大量连接分成较小的子集，使用有效的轮询技术来确定哪些连接已就绪并具有请求，以及哪些连接由于处于空闲状态的时间较长而被视为已关闭 （超过允许的保持活动超时的最大值）。

HTTP 侦听器的服务器名就是在重定向过程中由服务器发送给客户机的 URL 中使用的主机名。此属性会影响服务器自动生成的 URL，但不会影响存储在服务器中的目录和文件的 URL。如果服务器使用别名，则此名称通常为别名。如果客户机发送了一个 `Host:` 标题，则在重定向中该主机名将取代 HTTP 侦听器的服务器名。

要使用不同于原始请求中指定的端口号的端口号，请指定重定向端口。如果出现以下某一种情况，则会发生**重定向**：

- 如果客户机尝试访问已不存在于指定 URL 处的资源 （即该资源已被移动到其他位置），服务器将返回一个指定的响应代码，并在响应的位置标头中包含新的位置，从而将客户机重定向到新位置 （而不是返回 404）。
- 如果客户机尝试通过常规 HTTP 端口访问受保护 （例如 SSL）的资源，则服务器会将此请求重定向到启用了 SSL 的端口上。在此情况下，服务器将在位置响应标头中返回一个新的 URL，其中的原始非安全性端口已被替换为启用了 SSL 的端口。客户机随后将连接到这个新的 URL。

此外，还应指定是否为 HTTP 侦听器启用安全性以及使用哪种类型的安全性 （如使用哪一个 SSL 协议以及哪些加密算法）。

要访问部署在 Application Server 上的 Web 应用程序，请使用 URL `http://localhost:8080/`（或者，如果是安全应用程序，则使用 `http://localhost:8181/`）和为此 Web 应用程序指定的上下文根路径。要访问管理控制台，请使用 URL `https://localhost:4849/` 或 `https://localhost:4849/asadmin/`（其默认上下文根路径）。

由于虚拟服务器必须指定一个现有的 HTTP 侦听器，并且不能指定其他虚拟服务器已使用的 HTTP 侦听器，因此在创建新的虚拟服务器之前，应至少创建一个 HTTP 侦听器。

有关 HTTP 服务的管理控制台任务

配置 HTTP 服务

要配置 HTTP 服务，请执行以下步骤：

1. 在树组件中，展开“配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例（使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 选择“HTTP 服务”节点。
4. 在“HTTP 服务”页面中，您可以设置应用到该服务的所有 HTTP 侦听器的属性。

下表列出了这些属性。

表 17-1 HTTP 服务属性

属性名称	说明	默认值
traceEnabled	如果设置为 True，则启用 TRACE 操作。将此属性设置为 False 将使 Application Server 较少地受到跨站点脚本攻击的影响。	错误
monitoringCacheEnabled	如果设置为 True， Application Server 将高速缓存 HTTP 服务的统计数据的本地值，以响应统计数据查询。此值可用于提高性能。 如果设置为 False，Application Server 将查询 HTTP 服务的每个统计数据值。	TRUE
monitoringCacheRefreshInMillis	指定刷新监视高速缓存之前的时间间隔 （以毫秒为单位）。	5000
sslCacheEntries	指定可以高速缓存的 SSL 会话的数目。其值没有上限。	10000
sslSessionTimeout	指定 SSL2 会话超时之前的秒数。	100
ssl3SessionTimeout	指定 SSL3 会话超时之前的秒数。	86400
sslClientAuthDataLimit	指定客户机证书握手阶段缓冲的应用程序数据的最大量 （以字节为单位）。	1048576
sslClientAuthTimeout	指定客户机证书握手阶段超时之前的秒数。	60
keepAliveQueryMeanTime	指定所需的保持活动等待时间 （以毫秒为单位）。	100
keepAliveQueryMaxSleepTime	指定轮询保持活动连接的进一步请求之后休眠时间的上限 （以毫秒为单位）。	100
stackSize	指定本机线程的最大堆栈大小。	随操作系统 / 计算机而定
statsProfilingEnabled	如果设置为 False，将禁用 HTTP 服务的监视统计数据的记录 （可以提高性能）。如果将此属性设置为 False，则启用 HTTP 服务的监视功能不会有任何作用。	TRUE
chunkedRequestBufferSize	指定无块请求数据的默认缓冲区大小 （以字节为单位）。	8192
chunkedRequestTimeoutSeconds	指定无块请求数据的默认超时 （以秒为单位）。	60
dnsCacheEnabled	如果设置为 True，则允许用户监视与 DNS 高速缓存相关的统计信息。仅当选中 “HTTP 协议” 选项卡中的 “DNS 查找” 框时该属性才有效。否则，将忽略该属性设置。	错误

5. 单击 “访问日志”选项卡以配置访问日志轮转。单击其他选项卡以配置请求处理、保持活动子系统、连接池、 HTTP 协议和 HTTP 文件高速缓存。
6. 单击 “保存”。

配置 HTTP 服务访问日志

使用此页面可以为虚拟服务器启用和配置访问日志轮转。这些日志位于 *domain_root_dir*/*domain_dir*/logs/access 目录中，其命名如下：

*virtual_server_name*_access_log.*yyyy-mm-dd*.txt

单击 “装入默认值”可以装入默认值。要更改这些日志的轮转属性，请执行以下操作：

- 选中 “文件轮转”框以启用文件轮转。默认情况下，将启用文件轮转。
- 从 “轮转策略”下拉式列表中选择策略。（唯一可用的策略为 time。）
- 在 “轮转间隔”字段中，键入数值以指定访问日志轮转之间的分钟数。仅当轮转策略为 time 时，此字段才有效。默认值为 1440 分钟。
- 在 “轮转后缀”字段中，键入字符串值以指定轮转后添加到日志文件名中的后缀。默认值为 %YYYY;%MM;%DD;-%hh;h%mm;m%ss;s。
- 在 “格式”字段中，输入字符串值以指定访问日志的格式。使用下表中显示的格式。默认格式为 %client.name% %auth-user-name% %datetime% %request% %status% %response.length%。

表 17-2 访问日志格式的令牌值

数据	令牌
客户机主机名	%client.name%
客户机 DNS	%client.dns%
系统日期	%datetime%
完整 HTTP 请求行	%request%
状态	%status%
响应内容长度	%response.length%
引用者标头	%header.referer%
用户代理	%header.user-agent%

表 17-2 访问日志格式的令牌值 (续)

数据	令牌
HTTP 方法	%http-method%
HTTP URI	%http-uri%
HTTP 查询字符串	%query-str%
HTTP 协议版本	%http-version%
接受标头	%header.accept%
数据标头	%header.date%
If-Modified-Since 标头	%header.if-mod-since%
授权标头	%header.auth%
按照 RFC 2616 定义的任何有效 HTTP 标头值 （any 也是有效的标头值；此处将其指定为变量）	%header.*any*%
授权用户的名字	%auth-user-name%
Cookie 的值	%cookie.value%
虚拟服务器 ID	%vs.id%

- 单击 “保存”以保存更改，或者单击 “装入默认值”以返回到默认设置。

配置 HTTP 服务请求处理线程

使用此页面可以为 HTTP 服务配置请求处理线程：

- 单击 “装入默认值”可以装入默认值。
- 在 “线程计数”字段中，键入数值以指定请求处理线程的最大数目。默认值为 128。
- 在 “初始线程计数”字段中，键入服务器启动时可用的请求处理线程的数目。默认值为 48。
- 在 “线程增量”字段中，键入请求数目超过初始线程计数时要添加的请求处理线程的数目。默认值为 10。

- 在 “请求超时” 字段中，键入请求超时之前的秒数。默认值为 30 秒。
- 在 “缓冲区长度” 字段中，键入请求处理线程用于读取请求数据的缓冲区的大小 （以字节为单位）。默认值为 4096 字节。
- 单击 “保存” 以保存更改，或者单击 “装入默认值” 以返回到默认设置。

配置 HTTP 服务的保持活动子系统

使用此页面可以配置 HTTP 服务的保持活动子系统。

- 单击 “装入默认值” 可以装入默认值。
- 在 “线程计数” 字段中，键入要使用的保持活动线程的数目。默认值为 1。
- 在 “最大连接数” 字段中，键入要维护的持久性连接的最大数目。默认值为 256。
- 在 “超时” 字段中，键入保持活动连接应保持打开状态的最大秒数。默认值为 30 秒。
- 单击 “保存” 以保存更改，或者单击 “装入默认值” 以返回到默认设置。

另请参见：

- HTTP 侦听器

配置 HTTP 服务连接池

使用此页面可以为 HTTP 服务连接队列配置连接池：

- 单击 “装入默认值” 可以装入默认值。
- 在 “最大未决计数” 字段中，键入 HTTP 侦听器允许的最大未决连接数目。默认值为 4096。
- 在 “队列大小” 字段中，键入连接队列大小的最大值 （以字节为单位）。此值还指定服务器可以拥有的最大未完成连接数目。默认值为 4096。
- 在 “接收缓冲区大小” 字段中，为 HTTP 侦听器键入接收缓冲区的大小。默认值为 4096。
- 在 “发送缓冲区大小” 字段中，为 HTTP 侦听器键入发送缓冲区的大小。默认值为 8192。
- 单击 “保存” 以保存更改，或者单击 “装入默认值” 以返回到默认设置。

为 HTTP 服务配置 HTTP 协议

使用此页面可以为 HTTP 服务配置 HTTP 协议：

- 单击 “装入默认值”可以装入默认值。
- 在 “版本”字段中，键入要使用的 HTTP 协议的版本 （`HTTP/1.0` 或 `HTTP/1.1`）。默认值为 `HTTP/1.1`。
- 选中 “DNS 查找”框以启用客户机 DNS 条目的查找。默认值为 False。
- 取消选中 "SSL" 框可以全局禁用服务器中的安全性。将此值设置为 True 可以将 SSL 用于任何启用了安全性的侦听器。默认值为 True。
- 在 “强制响应类型”字段中，键入没有与扩展名匹配的可用 MIME 映射时使用的响应类型。默认值为 `text/html; charset=iso-8859-1`。
- 在 “默认响应类型”字段中，键入默认响应类型。默认值为 `text/html; charset=iso-8859-1`。此值是以分号分隔的字符串，由内容类型、编码、语言和字符集组成。
- 单击 “保存”以保存更改，或者单击 “装入默认值”以返回到默认设置。

为 HTTP 服务配置 HTTP 文件高速缓存

使用此页面可以为 HTTP 服务配置 HTTP 文件高速缓存。

文件高速缓存存储静态内容，以便服务器可以快速处理此类内容的请求。

- 单击 “装入默认值”可以装入默认值。
- 选中 “全局”框可以启用文件高速缓存。默认值为 True。
- 选中 “文件传输”框可以在 Windows 中启用 `TransmitFileSystem` 方法。默认值为 False。
- 在 “最长生存期”字段中，键入有效高速缓存条目的最长生存期 （以秒为单位）。默认值为 30 秒。
- 在 “最大文件计数”字段中，键入文件高速缓存中的最大文件数目。默认值为 1024。
- 在 “散列初始大小”字段中，键入散列桶的初始数目。默认值为零。

- 在 “中等文件大小限制” 字段中，键入可高速缓存为内存映射文件的文件大小的最大值 （以字节为单位）。默认值为 537,600 字节。
- 在 “中等文件大小” 字段中，键入高速缓存为内存映射文件的所有文件大小的总值 （以字节为单位）。默认值为 10,485,760 字节。
- 在 “小文件大小限制” 字段中，键入可读入内存的文件大小的最大值 （以字节为单位）。默认值为 2048 字节。
- 在 “小文件大小” 字段中，键入读入内存的所有文件大小的总值 （以字节为单位）。默认值为 1,048,576 字节。
- 从 “已启用文件高速缓存” 下拉式列表中选择 ON 或 OFF，以在文件大小小于中等文件大小限制时启用或禁用文件内容的高速缓存。默认值为 ON。
- 单击 “保存” 以保存更改，或者单击 “装入默认值” 以返回到默认设置。

有关虚拟服务器的管理控制台任务

创建虚拟服务器

要创建虚拟服务器，请执行以下步骤：

1. 在树组件中，展开 “配置” 节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要为将来的实例 （使用 default-config 的副本）配置默认设置，请选择 default-config 节点。

3. 展开 “HTTP 服务”节点。
4. 选择 “虚拟服务器”节点。
5. 在 “虚拟服务器”页面中，单击 “新建”。将显示 “创建虚拟服务器”页面。
6. 在 "ID" 字段中，为虚拟服务器键入唯一的名称。此值用于内部标识虚拟服务器，而不会提供给 HTTP 客户机。提供给 HTTP 客户机的主机名必须在 “主机”字段中指定。
7. 在 “主机”字段中，键入主机名或运行服务器的计算机名。请使用已向网络的 DNS 服务器注册 （对于 UNIX 系统，则是已在 `/etc/hosts` 文件中注册）的实际主机名或虚拟主机名。
8. 在对应的 “状态”区域中，选择 “打开”、“关闭”或 “禁用”。默认值为 “打开”。
9. 将 “HTTP 侦听器”字段保留为空白。在创建 HTTP 侦听器并将其与此服务器关联时，系统将自动填写此字段。

   （使用此字段时，需要指定现有的 HTTP 侦听器。但是，您不能指定由其他虚拟服务器使用的侦听器；否则，服务器日志中将显示错误。由于侦听器在创建时必须与一个现有的虚拟服务器相关联，因此所有的现有侦听器均已被其他虚拟服务器使用。）
10. 在 “默认 Web 模块”下拉列表中，选择已部署的 Web 模块 （如果有），此模块将响应所有无法映射到已部署到虚拟服务器的其他 Web 模块的请求。

    如果不指定默认 Web 模块，系统将使用具有空上下文根路径的 Web 模块。如果不存在包含空上下文根的 Web 模块，将创建一个系统默认模块并使用它。
11. 在 “日志文件”字段中，键入将记录此虚拟服务器的日志信息的文件的路径名。如果要将日志信息发送到默认服务器日志 *domain_root_dir*/*domain_dir*/`logs/server.log`，请将此字段保留为空。
12. 在 “其他属性”区域中，单击 “添加属性”以为虚拟服务器添加属性。无论是否指定属性，新的服务器都会将默认属性 `docroot` 和 `accesslog` 设置为默认值。
13. 单击 “确定”以保存该虚拟服务器。

下表列出了可用的属性。

表 17-3 虚拟服务器属性

属性名称	说明
docroot	服务器的文档根目录的绝对路径。 默认值为 *domain_root_dir*/*domain_dir*/docroot。
accesslog	服务器访问日志的绝对路径。 默认值为 *domain_root_dir*/*domain_dir*/logs/access。
sso-enabled	如果设置为 False，则此虚拟服务器将禁用单点登录，用户必须在使用虚拟服务器上的每个应用程序时都分别进行验证。 Servlet 和 JSP 页面支持在 Application Server 上跨应用程序的单点登录。此功能允许需要同一用户登录信息的多个应用程序共享登录信息，用户不必在使用每个应用程序时都分别进行登录。 默认值为 True。
sso-max-inactive-seconds	指定如果未接收到任何客户机活动，在清除用户的单点登录记录前等待的时间 （以秒为单位）。由于单点登录应用到同一个虚拟服务器上的多个应用程序，因此对其中任何一个应用程序的访问都可以使单点登录记录保持活动状态。 默认值为 300 秒 （5 分钟）。较高的值为用户提供了较长的单点登录持久性，但会占用服务器的更多内存。
sso-reap-interval-seconds	指定清除到期的单点登录记录的时间间隔 （以秒为单位）。 默认值为 60。

allowLinking	如果设置为 True，则将为部署到该虚拟服务器上的所有 Web 应用程序提供符号链接资源。通过使用 sun-web.xml 文件中的 sun-web-app 属性 allowLinking，各个 Web 应用程序可以覆盖该设置。
	`<sun-web-app>`
	`  <property name="allowLinking" value="{true|false}"/>`
	`</sun-web-app>`
	默认值为 True。

等效的 asadmin 命令为：create-virtual-server

编辑虚拟服务器

要编辑虚拟服务器，请执行以下步骤：

1. 在树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 展开 “HTTP 服务”节点。
4. 选择 “虚拟服务器”节点。
5. 选择要编辑的虚拟服务器。
6. 在 “编辑虚拟服务器”页面中，您可以执行以下这些任务：
   - 在 “主机”字段中更改主机名。
   - 更改 “状态”设置的值。
   - 添加或删除 HTTP 侦听器。
   - 更改 “默认 Web 模块”的选择。
   - 更改 “日志文件”的值。
   - 添加、删除或修改属性。
7. 单击 “保存”以保存更改。

删除虚拟服务器

要删除虚拟服务器，请执行以下步骤：

1. 在树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。

3. 展开 “HTTP 服务”节点。
4. 选择 “虚拟服务器”节点。
5. 在 “虚拟服务器”页面中，选中要删除的虚拟服务器名称旁边的框。
6. 单击 “删除”。

可以删除 __asadmin 虚拟服务器，但是建议不要这样做。如果打算这样做，请先将 Application Server 的 domain.xml 文件中的 virtual-server 元素复制到安全的位置，以便可以在需要时恢复这些设置。

等效的 asadmin 命令为：delete-virtual-server

有关 HTTP 侦听器的管理控制台任务

创建 HTTP 侦听器

要创建 HTTP 侦听器，请执行以下步骤：

1. 在树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要为将来的实例 （使用 default-config 的副本）配置默认设置，请选择 default-config 节点。
3. 展开 “HTTP 服务”节点。
4. 选择 “HTTP 侦听器”节点。
5. 在 “HTTP 侦听器”页面中，单击 “新建”。将显示 “创建 HTTP 侦听器”页面。
6. 在 “名称”字段中，键入侦听器的名称。

7. 如果您不希望在服务器重新启动时启用侦听器，请在 “侦听器”字段中取消选中 “已启用”框。
8. 在 “网络地址”字段中，如果希望侦听器侦听此服务器的所有 IP 地址，请键入 0.0.0.0 并使用唯一的端口值。否则，请键入此服务器的有效 IP 地址。
9. 在 “侦听器端口”字段中，如果 “网络地址”字段为 0.0.0.0，请键入唯一的端口值，或者，如果要使用其他 IP 地址，请键入所需的端口值。
10. 从 “默认虚拟服务器”下拉列表中选择一个虚拟服务器。
11. 在 “服务器名”字段中，键入服务器发送给客户机的 URL 中使用的主机名。如果服务器使用别名，则此名称为别名。如果服务器未使用别名，请将此字段保留为空。
12. 在 “高级”区域中，可以执行以下任务：
    - 要将请求重定向到其他端口，请在 “重定向端口”字段中键入一个值。如果满足以下两个条件， Application Server 将自动重定向请求：
      - 此侦听器支持非 SSL 请求。
      - 接收匹配安全性约束需要 SSL 传输的请求。

      默认情况下， Application Server 使用原始请求中指定的端口号。
    - 更改接收器线程的数目。
    - 取消选中 “支持者”框以在 Servlet 生成的 HTTP 响应标头中禁用 `X-Powered-By:Servlet/2.4` 标头。

      Java Servlet 2.4 规范中定义了此标头，其容器可以添加到 Servlet 生成的响应。类似地， JavaServer Pages™(JSP™) 2.0 规范中定义了一个 `X-Powered-By:JSP/2.0` 标头，此标头将添加到使用 JSP 技术的响应 （可选）。默认情况下，将为 Web 应用程序包含 `X-Powered-By:JSP/2.0` 标头。这些标头的目的是帮助 Web 站点管理员收集有关 Servlet 和 JSP 技术使用方面的统计数据。

      有关启用和禁用 JSP 页面的 `X-Powered-By` 标头的信息，请参见《Application Server Developer's Guide》中的 "Deployment Descriptor Files" 一章。请参见第 48 页上的 “更多信息”以获得指向此文档的链接。

      根据生产环境的不同，可以省略 `X-Powered-By` 标头的生成，以隐藏基本技术。
13. 要创建不启用安全性的侦听器，请单击 “确定”。

在此页面的 "SSL" 部分中，您可以将侦听器配置为使用 SSL 安全性、 TLS 安全性或同时使用 SSL 和 TLS 安全性。

要设置安全侦听器，请执行以下操作：

1. 在 “安全性”字段中选中 “启用”框。
2. 要强制客户机在使用此侦听器时自行向服务器进行验证，请在 “客户机验证”字段中选中 “已启用”框。
3. 在 “证书昵称”字段中输入现有服务器密钥对和证书的名称。有关详细信息，请参见 “安全性”一章。
4. 在 "SSL3/TLS" 部分：
   a. 选取要在侦听器上启用的安全性协议。选取 "SSL3"、 "TLS" 或两者。
   b. 选取协议所使用的加密算法套件。要启用所有加密算法套件，请选取 “所有支持的加密算法套件”。您还可以启用单个加密算法套件。

此时，侦听器将在已指定为默认虚拟服务器的虚拟服务器的 “HTTP 侦听器”字段中列出。

等效的 `asadmin` 命令为：`create-http-listener`、 `create-ssl`

编辑 HTTP 侦听器

要编辑 HTTP 侦听器，请执行以下步骤：

1. 在树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 展开 “HTTP 服务”节点。
4. 选择 “HTTP 侦听器”节点。
5. 选择要编辑的 HTTP 侦听器。
6. 在 “编辑 HTTP 侦听器”页面中，修改其中的设置。
7. 单击 “保存”以保存更改。

删除 HTTP 侦听器

要删除 HTTP 侦听器，请执行以下步骤：

1. 在树组件中，展开 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 展开 “HTTP 服务”节点。
4. 选择 “HTTP 侦听器”节点。
5. 在 “HTTP 侦听器”页面中，选中要删除的 HTTP 侦听器名称旁边的框。
6. 单击 “删除”。

可以删除 `http-listener-1`、`http-listener-2` 和 `admin-listener` HTTP 侦听器，但是最好不要这样做。如果打算这样做，请先将 Application Server 的 `domain.xml` 文件中的 `http-listener` 元素复制到安全的位置，以便可以在需要时恢复这些设置。

等效的 `asadmin` 命令为：`delete-http-listener`

第 18 章

配置对象请求代理

本章介绍了如何配置对象请求代理 (ORB) 和 IIOP 侦听器。本章包含以下几节：

关于对象请求代理

CORBA

Application Server 支持标准的协议和格式集来确保互操作性。这些协议之间的协议是由 CORBA 定义的。

CORBA （公共对象请求代理体系结构）模型的基础是：客户机以远程方法请求形式向分布式对象或服务器发出请求，并通过明确定义的接口从这些对象那里请求服务。远程方法请求携带了有关需要执行的操作的信息，其中包括被调用方法的服务提供商的对象名称 （称为对象引用）和参数 （如果有）。 CORBA 自动处理网络程序任务，如对象注册、对象定位、对象激活、请求多路复用、错误处理、编组和操作分发。

什么是 ORB？

对象请求代理 (ORB) 是 CORBA 的核心组件。ORB 提供了识别和定位对象、处理连接管理、传送数据和请求通信所需的框架结构。

CORBA 对象之间从不直接进行通信，对象通过远程桩对运行在本地计算机上的 ORB 发出请求。本地 ORB 使用 Internet Inter-Orb 协议 （IIOP 为缩写形式）将该请求传递给其他计算机上的 ORB。然后，远程 ORB 定位相应的对象、处理该请求并返回结果。

使用 RMI-IIOP，应用程序或对象可将 IIOP 用作远程方法调用 (RMI) 协议。企业 Bean （EJB 模块）的远程客户机通过 RMI-IIOP 与 Application Server 进行通信。

IIOP 侦听器

IIOP 侦听器是一个侦听套接字，它接收来自企业 Bean 的远程客户机和其他基于 CORBA 的客户机的外来连接。可以为 Application Server 配置多个 IIOP 侦听器。为每个侦听器指定一个端口号、一个网络地址和 （可选）多个安全性属性。有关更多信息，请参见第 307 页上的 “创建 IIOP 侦听器”。

用于 ORB 的管理控制台任务

配置 ORB

1. 在树组件中，展开 “配置” 节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 选择 "ORB" 节点。

4. 从 “线程池 ID” 下拉式列表中选择 ORB 使用的线程池。

   ORB 使用线程池响应来自通过 RMI-IIOP 进行通信的企业 Bean 的远程客户机和其他客户机的请求。有关更多信息，请参见第 311 页上的 “Application Server 中的线程池”和第 312 页上的 “创建线程池”。

5. 在 “最大消息片段大小” 字段中，设置 IIOP 消息的最大片段大小。

   大于此大小的消息将被分段。

6. 在 “连接总数” 字段中，设置所有 IIOP 侦听器的最大外来连接数。

7. 如果需要进行 IIOP 客户机验证，请选中 “需要” 复选框。

8. 单击 “保存” 以保存更改，或者单击 “装入默认值” 以装入默认值。

9. 重新启动服务器。

用于 IIOP 侦听器的管理控制台任务

创建 IIOP 侦听器

1. 在树组件中，展开 “配置” 节点。

2. 选择要配置的实例：

   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。

   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。

3. 展开 "ORB" 节点。

4. 选择 “IIOP 侦听器”。

5. 单击 “新建”。

6. 在 “名称” 字段中输入用于标识侦听器的名称。

7. 在 “网络地址”字段中输入侦听器的网络地址。

   此地址可以是 IP 地址，也可以是 DNS 可解析的主机名。

8. 在 “侦听器端口”字段中，输入侦听器要进行侦听的端口号。

9. 在 “侦听器”字段中，选中 “已启用”框以启用侦听器。

10. 在 “其他属性”区域中，为应用程序所需的属性提供值。

11. 要创建不启用安全性的侦听器，请单击 “确定”。

在此页面的 “安全性”部分中，您可以将侦听器配置为使用 SSL 安全性、 TLS 安全性或同时使用 SSL 和 TLS 安全性。

要设置安全侦听器，请执行以下操作：

1. 在 “安全性”字段中选中 “启用”框。

2. 要强制客户机在使用此侦听器时自行向服务器进行验证，请在 “客户机验证”字段中选中 “已启用”框。

3. 在 “证书昵称”字段中输入现有服务器密钥对和证书的名称。

4. 在 "SSL3/TLS" 部分：

   a. 选取要在侦听器上启用的安全协议。选取 "SSL3" 或 "TLS"，或同时启用这两种协议。

   b. 选取协议所使用的加密算法套件。要启用所有加密算法套件，请选取 “所有支持的加密算法套件”。您还可以启用单个加密算法套件。

5. 单击 “确定”。

现在，该侦听器将列在 “IIOP 侦听器”页面的 “当前侦听器”表中。

等效的 `asadmin` 命令为：`create-iiop-listener`、 `create-ssl`

编辑 IIOP 侦听器

1. 在树组件中，展开 “配置” 节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 展开 "ORB" 节点。
4. 选择 “IIOP 侦听器” 节点。
5. 在 “当前侦听器” 表中选择要修改的侦听器。
6. 修改该侦听器的设置。有关可修改字段的说明，请参见第 307 页上的 “创建 IIOP 侦听器”。
7. 如果已更改侦听器的端口号，请重新启动服务器。

删除 IIOP 侦听器

1. 在树组件中，展开 “配置” 节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 展开 "ORB" 节点。
4. 选择 “IIOP 侦听器” 节点。
5. 在 “当前侦听器” 表中选取要删除的侦听器。
6. 单击 “删除”。

等效的 `asadmin` 命令为：`delete-iiop-listener`

第 19 章

线程池

本章介绍如何创建、编辑和删除线程池。它包含以下各节：

- 关于线程池
- 有关线程池的管理控制台任务

关于线程池

本节介绍线程池及其在 Application Server 中的工作方式。

Application Server 中的线程池

Java 虚拟机 (JVM) 可以支持一次执行多个线程。为了提高性能，Application Server 维护了一个或多个线程池。可以将特定的线程池指定给连接器模块和 ORB。

一个线程池可以提供多个连接器模块和企业 Bean。请求线程处理对应用程序组件的用户请求。服务器接收到请求时，它会将请求指定给线程池中的空闲线程。该线程执行客户机的请求并返回结果。例如，如果请求需要使用的系统资源当前正处于忙碌状态，则线程会在允许请求使用该资源前，等待资源回到空闲状态。

指定为来自应用程序的请求预留的最大线程数和最小线程数。线程池在这两个值之间动态调整。指定的最小线程池大小将通知服务器为应用程序请求至少分配该大小的预留线程数。可以将线程数增加到所指定的最大线程池大小。

如果增加可供进程使用的线程数，则该进程可以同时对更多的应用程序进行响应。

通过将 Application Server 线程分到不同的线程池中，可以避免在一个资源适配器或应用程序占用 Application Server 中的所有线程时出现线程不足的情况。

有关线程池的管理控制台任务

创建线程池

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要配置所有实例的默认设置，请选择 `default-config` 节点。
3. 选择 “线程池”节点。
4. 在 “当前池”下单击 “新建”。
5. 在 “线程池 ID”字段中输入线程池的名称。
6. 在 “最小线程池大小”字段中，输入服务此队列中的请求的线程池中线程的最小数目。

   将此线程池实例化时将预先创建这些线程。
7. 在 “最大线程池大小”字段中，输入服务此队列中的请求的线程池中线程的最大数目。

   这是存在于此线程池中的线程数上限。
8. 在 “空闲超时”字段中输入数值 （以秒为单位），超过此时间段之后将从池中删除空闲线程。
9. 在 “工作队列的数目”字段中，输入由此线程池服务的工作队列的总数。
10. 单击 “确定”。
11. 重新启动 Application Server。

等效的 `asadmin` 命令为：`create-threadpool`

编辑线程池

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要配置所有实例的默认设置，请选择 default-config 节点。
3. 选择 “线程池”节点。
4. 在 “当前池”下选择要更改的线程池的名称。
5. 在 “最小线程池大小”字段中，输入服务此队列中的请求的线程池中线程的最小数目。

   将此线程池实例化时将预先创建这些线程。
6. 在 “最大线程池大小”字段中，输入服务此队列中的请求的线程池中线程的最大数目。

   这是存在于此线程池中的线程数上限。
7. 在 “空闲超时”字段中输入数值 （以秒为单位），超过此时间段之后将从池中删除空闲线程。
8. 在 “工作队列的数目”字段中，输入由此线程池服务的工作队列的总数。
9. 单击 “保存”。
10. 重新启动 Application Server。

删除线程池

1. 在树组件中，选择 “配置”节点。
2. 选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要配置所有实例的默认设置，请选择 default-config 节点。

3. 选择 “线程池”节点。
4. 在 “当前池”表中选取要删除的线程池名称。
5. 单击 “删除”。
6. 重新启动 Application Server。

等效的 asadmin 命令为：delete-threadpool

第 20 章

配置日志记录

本章简要介绍了如何使用管理控制台来配置日志记录和查看服务器日志。它包含以下小节：

关于日志

日志记录

Application Server 使用在 JSR 047 中指定的 Java 2 平台日志记录 API。Application Server 日志信息记录在服务器日志中，您通常可以在 *domain_root_dir*/*domain_dir*/`logs/server.log` 中找到该日志。

domain_root_dir/*domain_dir*/`logs/` 目录中除了包含服务器日志外，还包含另外两种日志。`access` 子目录中包含 HTTP 服务访问日志，`tx` 子目录中包含事务服务日志。有关这些日志的信息，请参见第 293 页上的 “配置 HTTP 服务访问日志”和第 283 页上的 “配置事务”。

Application Server 的组件生成日志记录输出。应用程序组件也可以生成日志记录输出。

应用程序组件可以使用 Apache Commons Logging Library 来记录消息。但是，建议采用平台标准 JSR 047 API 以获得更好的日志配置。

日志记录遵循以下统一格式：

[#|*yyyy-mm-ddThh:mm:ss.SSS-Z*|**日志级别**|**产品名称版本**|**日志程序名称**|**关键字值对**|**消息**|#]

例如：

```
[#|2004-10-21T13:25:53.852-0400|INFO|sun-appserver-ee8.1|javax.enterprise.
system.core|_ThreadID=13;|CORE5004: Resource Deployed:
[cr:jms/DurableConnectionFactory].|#]
```

在本示例中，

- `[#` 和 `#]` 标记该记录的开始和结束。
- 垂直条 (|) 用于分隔记录字段。
- `2004-10-21T13:25:53.852-0400` 指明了日期和时间。
- **日志级别**为 `INFO`。日志级别可以是以下任何值：`SEVERE`、 `WARNING`、 `INFO`、 `CONFIG`、 `FINE`、 `FINER` 和 `FINEST`。
- **产品名称版本**为 `sun-appserver-ee8.1`。
- **日志程序名称**是用于标识日志模块的资源的分层日志程序名称空间，在此示例中为 `javax.enterprise.system.core`。
- **关键字值对**为关键字名称和值，通常为线程 ID，如 `_ThreadID=14;`。
- **消息**是日志消息的文本。对于所有的 Application Server `SEVERE` 和 `WARNING` 消息以及多种 `INFO` 消息，它均以包含模块代码和数值的消息 ID 开头（在此示例中为 `CORE5004`）。

在以后的版本中，可能会更改或增强日志记录格式。

日志程序名称空间分层结构

Application Server 为它的每个模块都提供了日志程序。下表按照每个日志程序的模块名称和名称空间在管理控制台的 “日志级别”页面中的显示方式以字母顺序列出每个日志程序的模块名称和名称空间 （请参见第 319 页上的 “配置日志级别”）。“日志级别”页面中未显示表中最后三个模块。

表 20-1 Application Server 日志程序名称空间

模块名称	名称空间
管理	javax.enterprise.system.tools.admin
类加载器	javax.enterprise.system.core.classloading
CMP	javax.enterprise.system.container.cmp
配置	javax.enterprise.system.core.config
连接器	javax.enterprise.resource.resourceadapter
CORBA	javax.enterprise.resource.corba
部署	javax.enterprise.system.tools.deployment
EJB 容器	javax.enterprise.system.container.ejb
JavaMail	javax.enterprise.resource.javamail
JAXR	javax.enterprise.resource.webservices.registry
JAX-RPC	javax.enterprise.resource.webservices.rpc
JDO	javax.enterprise.resource.jdo
JMS	javax.enterprise.resource.jms
JTA	javax.enterprise.resource.jta
JTS	javax.enterprise.system.core.transaction
MDB 容器	javax.enterprise.system.container.ejb.mdb
命名	javax.enterprise.system.core.naming
节点代理 （仅限于 Enterprise Edition）	javax.ee.enterprise.system.nodeagent
根目录	javax.enterprise
SAAJ	javax.enterprise.resource.webservices.saaj
安全性	javax.enterprise.system.core.security
服务器	javax.enterprise.system
同步 （仅限于 Enterprise Edition）	javax.ee.enterprise.system.tools.synchronization
实用程序	javax.enterprise.system.util
验证器	javax.enterprise.system.tools.verifier
Web 容器	javax.enterprise.system.container.web
核心	javax.enterprise.system.core
系统输出 (System.out.println)	javax.enterprise.system.stream.out
系统错误 (System.err.println)	javax.enterprise.system.stream.err

用于日志记录的管理控制台任务

配置常规日志记录设置

1. 在树组件中，展开 “节点代理”节点或 “配置”节点。
2. 选择节点代理，或者选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 `server`，请选择 `server-config` 节点。
   b. 要为将来的实例 （使用 `default-config` 的副本）配置默认设置，请选择 `default-config` 节点。
3. 对于节点代理，请选择 “日志程序设置”选项卡。对于配置，请选择 “日志程序设置”节点。
4. 在 “日志设置”页面中，可以使用以下字段来自定义日志：
   - **日志文件**：要为服务器日志文件指定替代名称或位置，请在文本字段中键入新的路径名。默认情况下，该位置为 *domain_root_dir*`/`*domain_dir*`/logs/server.log`。
   - **警报**：要通过 JMX 框架路由 `SEVERE` 和 `WARNING` 消息，请选中 “已启用”复选框。
   - **写入系统日志**：仅在 Solaris 和 Linux 系统中，如果要将日志输出发送到服务器日志和 `syslog` 工具，请选中 “已启用”复选框。
   - **日志处理程序**：要将日志发送到 `server.log` 或 `syslog` 以外的目标，您可以插入自定义日志处理程序。自定义处理程序必须扩展 `java.util.logging.Handler` 类 （JSR 047 兼容 API）。在 “日志处理程序”字段中键入处理程序的绝对类名。还应将处理程序类置于 Application Server 类路径中，以便在服务器启动过程中安装该处理程序。自定义处理程序的日志记录具有在第 315 页上的 “日志记录”中说明的格式。

- **日志过滤器**：要过滤发送给目标 （如 server.log、 syslog 或由自定义日志处理程序指定的目标）的日志记录，可以插入自定义日志过滤器。该自定义过滤器必须实现接口 java.util.logging.Filter。在 “日志过滤器” 字段中键入过滤器的绝对类名。还应将过滤器类置于 Application Server 类路径中，以便在服务器启动过程中安装该过滤器。
- **文件轮转限制**：如果服务器日志的字节数达到了指定大小，请创建一个名为 server.log 的新的空文件，并将旧文件重命名为 server.log_*date*，其中 *date* 是轮转文件的日期和时间。默认值为 2 兆字节。该限制的最小值为 500 千字节；如果指定较低的值，则要在达到 500 千字节时该文件才轮转。要关闭日志文件轮转，请将该值设置为 0。
- **文件轮转时间限制**：达到指定的分钟数之后才轮转服务器日志。默认值为零，这表示文件达到 “文件轮转限制” 字段中指定的大小即被轮转。**如果指定了非零的时间限制，则该时间限制优先于大小限制**。

5. 单击 “保存” 以保存更改。单击 “查看日志文件” 查看服务器日志。

配置日志级别

1. 在树组件中，展开 “节点代理” 节点或 “配置” 节点。
2. 选择节点代理，或者选择要配置的实例：
   a. 要配置特定的实例，请选择该实例的配置节点。例如，对于默认实例 server，请选择 server-config 节点。
   b. 要为将来的实例 （使用 default-config 的副本）配置默认设置，请选择 default-config 节点。
3. 对于节点代理，请选择 “日志级别” 选项卡。对于配置，请选择 “日志程序设置” 节点，然后选择 “日志级别” 选项卡。
4. 在 “模块日志级别” 页面中，请从要更改日志级别的一个或多个模块对应的下拉式列表中选择一个新值。默认级别为 INFO，表示处于该级别或更高级别 （WARNING、 SEVERE）的消息将显示在日志中。可以选择以下任一值 （以最高级别至最低级别的顺序列出）：
   - SEVERE
   - WARNING
   - INFO

- CONFIG
- FINE
- FINER
- FINEST
- OFF

5. 使用 “其他属性”区域可以为任何应用程序日志程序配置日志级别。属性名是日志程序名称空间，其值为上述八个可能的级别之一。例如，属性名称可为 samples.logging.simple.servlet，值可为 FINE。

   还可以使用此区域来更改子模块的日志级别，例如 CORBA 模块的传输子模块：

   javax.enterprise.resource.corba.*ORBId*.transport

6. 单击 “保存”以保存更改，或者单击 “装入默认值”以恢复默认值。

对 System.out.println 的调用均在 INFO 级别使用日志程序名称 javax.enterprise.system.stream.out 予以记录。对 System.err.println 的调用均在 WARNING 级别使用日志程序名称 javax.enterprise.system.stream.err 予以记录。要从这些源关闭日志，请在 “其他属性”区域中将日志程序名称的值指定为 OFF。

对日志级别设置所做的更改将立即生效。同时将在 domain.xml 文件中保存这些更改，以供服务器重新启动时使用。

查看服务器日志

1. 在树组件中，展开要查看其日志的服务器实例的节点。
2. 在 “常规信息”页面中，单击 “查看日志文件”。

使用 “搜索条件”区域自定义和过滤日志查看器。使用如下所示的基本字段：

- **实例名称**：从下拉式列表中选择一个实例名称，以查看该服务器实例的日志。默认值为当前服务器实例。
- **日志文件**：从下拉式列表中选择一个日志文件名称，以查看该日志的内容。默认名称为 server.log。

- **时间戳**：要查看最新的消息，请选择 “最近”（默认值）。如果只查看特定时间段内的消息，请选择 “特定范围”并在显示的 “从”字段和 “到”字段中键入日期和时间值。对于时间值，其语法必须采用以下格式 （*SSS* 表示毫秒）：

  hh:*mm*:*ss*.*SSS*

  例如：

  17:10:00.000

  如果 “从”字段中的时间值迟于 “到”字段中的时间值，将显示错误消息。

- **日志级别**：要按照日志级别过滤消息，请从下拉式列表中选择一种日志级别。默认情况下，将显示服务器日志中选定日志级别和更高日志级别的所有消息。选中标有 “不包括更高级别的消息”的复选框将仅显示所选级别的消息。

  为确保您要查看的消息都显示在服务器日志中，请先在 “日志级别”页面中设置适当的日志级别。请参见第 319 页上的 “配置日志级别”。

  如果您选择基于日志级别过滤日志消息，则将只显示符合指定过滤条件的消息。不过，这种过滤不影响那些记录到服务器日志中的消息。

将显示服务器日志中最新的 40 个条目以及在 “日志记录设置”和 “日志级别”页面中指定的设置。

单击 “时间戳”标题旁边的三角形对这些消息进行排序，以使最新的消息显示在最后。

要查看任何消息的格式化版本，请单击标记的链接

(details)

将显示标有 “日志条目详细资料”的窗口，该窗口包含了消息的格式化版本。

在条目列表的末尾，单击按钮以查看日志文件中较早或较晚的条目。

单击 “搜索条件”区域中的 “高级搜索”以进一步细化日志查看器的搜索条件。使用如下所示的 “高级选项”字段：

- **日志程序**：要按模块过滤，请从下拉式列表中选择一个或多个名称空间。按住 Shift 键并单击或按住 Ctrl 键并单击来选择多个名称空间。

  选择较高级别的名称空间也就选择了该名称空间下的所有名称空间。例如，选择 javax.enterprise.system 的同时也就选择了该名称空间下所有模块的日志程序：javax.enterprise.system.core、javax.enterprise.system.tools.admin 等等。

- **自定义日志程序**：要查看特定于某个特定应用程序的日志程序中的消息，请在文本字段中键入日志程序的名称，每行键入一个。如果应用程序具有多个模块，您可以查看任何模块，也可以查看所有模块。例如，假定应用程序具有使用以下名称的日志程序：

  ```
  com.mycompany.myapp.module1
  com.mycompany.myapp.module2
  com.mycompany.myapp.module3
  ```

  要查看应用程序中的所有模块的消息，请键入 `com.mycompany.myapp`。如果只查看 `module2` 的消息，则键入 `com.mycompany.myapp.module2`。

  如果指定了一个或多个自定义日志程序，则仅当您在 “日志程序”区域中明确地指定 Application Server 模块的消息之后，才会显示这些消息。

- **名称 - 值对**：要查看特定线程的输出，请在文本字段中键入该线程的关键字名称和值。关键字名称为 `_ThreadID`。例如：

  ```
  _ThreadID=13
  ```

  假定 `com.mycompany.myapp.module2` 在多个线程中运行。要限制日志查看器使其只显示单个线程的输出，请在 “自定义日志程序”字段中指定该模块的日志程序，然后在此字段中指定线程 ID。

- **显示**：要一次查看 40 条以上的消息 （默认值），请从下拉式列表中选择其他可用的值 （100、 250 或 1000）。

  要查看堆栈追踪，请取消选中 “限制过长的消息”复选框。默认情况下，查看器中不会显示堆栈追踪，要查看堆栈追踪，请单击消息的 (`details`) 链接。

单击 “基本搜索”可以隐藏 “高级选项”区域。

第 21 章

监视组件和服务

本章包含有关使用 Application Server 管理控制台监视组件的信息。本章包含以下几节：

关于监视

监视 Application Server

使用监视功能可以观察 Sun Java System Application Server Enterprise Edition 8.1 2005Q1 的服务器实例中部署的各种组件和服务的运行时状态。利用有关运行时组件和进程状态的信息，可以确定性能瓶颈以便进行优化、有助于进行容量规划、预测故障、在发生故障时分析根本原因，以及确保一切运行正常。

启用监视功能会因增加系统开销而使性能降低。

监视概述

要监视 Application Server，请执行以下步骤：

1. 使用管理控制台或 asadmin 工具来启用对特定服务和组件的监视功能。

   有关此步骤的更多信息，请参阅 “有关启用和禁用监视功能的管理控制台任务”。

2. 使用管理控制台或 asadmin 工具来查看指定服务或组件的监视数据。

   有关此步骤的更多信息，请参阅 “有关查看监视数据的管理控制台任务”。

关于可监视对象的树结构

Application Server 使用树结构来跟踪可监视对象。由于监视对象的树是动态的，因此在实例中添加、更新或删除组件时该树会相应地发生变化。树中的根对象为服务器实例名称 （例如 server）。（在平台版中，仅允许使用一个服务器实例。）

以下命令显示了树的顶层：

```
asadmin> list --monitor server
server.applications
server.http-service
server.connector-service
server.jms-service
server.jvm
server.orb
server.resources
server.thread-pools
```

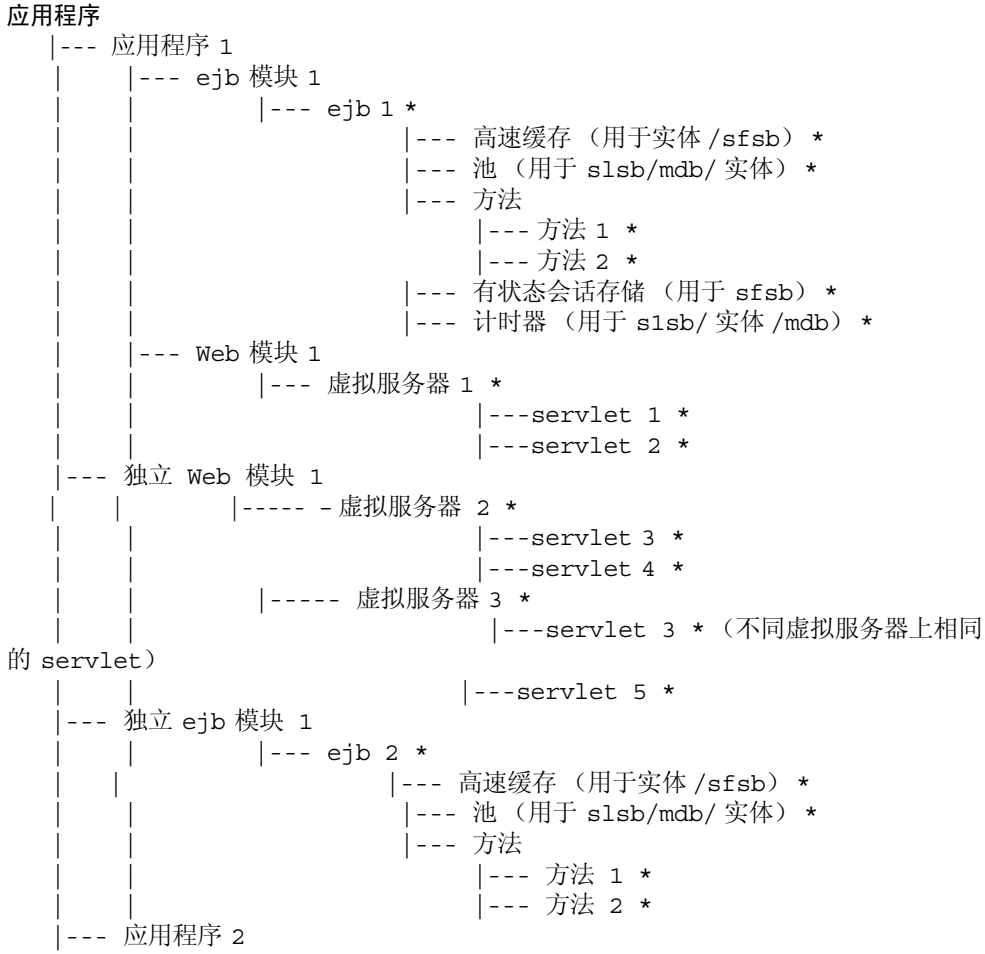
以下各节介绍了这些子树：

- 应用程序树
- HTTP 服务树
- 连接器服务树
- 连接器服务树
- JMS 服务树
- ORB 树
- 线程池树

应用程序树

以下示意图显示了企业应用程序的各种组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。有关更多信息，请参阅 “EJB 容器统计信息”和“Web 容器统计信息”。

图 21-1 应用程序节点树结构



HTTP 服务树

以下示意图显示了 HTTP 服务的节点。具有可用的监视信息的节点标有星号 (*)。请参见 “HTTP 服务树”。

图 21-2 HTTP 服务示意图 （PE 版）

```
http 服务
        |--- 虚拟服务器 1
           |--- http 侦听器 1 *
           |--- http 侦听器 2 *
        |--- 虚拟服务器 2
           |--- http 侦听器 1 *
           |--- http 侦听器 2 *
```

图 21-3 HTTP 服务示意图 （EE 版）

```
http 服务 *
            |--- 连接队列 *
            |--- dns *
            |--- 文件高速缓存 *
            |--- 保持活动 *
            |--- pwc 线程池 *
            |--- 虚拟服务器 1 *
            |          |--- 请求 *
            |--- 虚拟服务器 2 *
            |          |--- 请求 *
```

资源树

资源节点保存 JDBC 连接池、连接器连接池等池的可监视属性。以下示意图显示了各种资源组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。请参见 “JDBC 连接池统计信息”和 “JMS 连接器服务统计信息”。

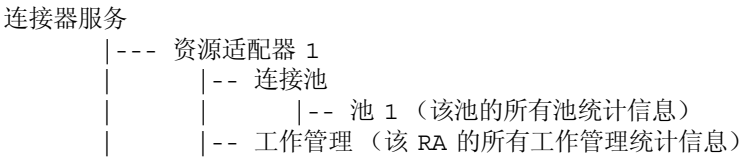
图 21-4 资源示意图

```
资源
       |--- 连接池 1 （连接器连接池或 jdbc） *
       |--- 连接池 2 （连接器连接池或 jdbc） *
```

连接器服务树

连接器服务节点保存连接器连接池等池的可监视属性。以下示意图显示了各种连接器服务组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。请参见 “JMS 连接器服务统计信息”。

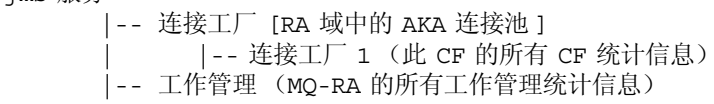
图 21-5 连接器服务示意图



JMS 服务树

JMS 服务节点保存连接器连接池等池的可监视属性。以下示意图显示了各种 JMS 服务组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。

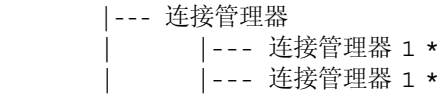
图 21-6 JMS 服务示意图



ORB 树

ORB 节点保存连接管理器的可监视属性。以下示意图显示了 ORB 组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。请参见 “ORB 中连接管理器的统计信息”。

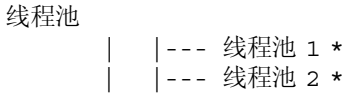
图 21-7 ORB 示意图



线程池树

线程池节点保存连接管理器的可监视属性。以下示意图显示了 ORB 组件的顶层节点和子节点。具有可用的监视统计信息的节点标有星号 (*)。请参见 “线程池统计信息”。

图 21-8 线程池示意图



关于受监视的组件和服务的统计信息

本节介绍了可用的监视统计信息：

- EJB 容器统计信息
- Web 容器统计信息
- HTTP 服务统计信息
- JDBC 连接池统计信息
- JMS 连接器服务统计信息
- ORB 中连接管理器的统计信息
- 线程池统计信息
- 事务服务统计信息
- Java 虚拟机 (JVM) 统计信息
  - J2SE 5.0 中的 JVM 统计信息
- 生产 Web 容器 (PWC) 统计信息

EJB 容器统计信息

表 21-1 中介绍了 EJB 统计信息。

表 21-1 EJB 统计信息

属性名称	数据类型	说明
createcount	计数 统计信息	调用 EJB 的 create 方法的次数。
removecount	计数 统计信息	调用 EJB 的 remove 方法的次数。
pooledcount	范围 统计信息	处于汇集状态的实体 Bean 的数目。
readycount	范围 统计信息	处于就绪状态的实体 Bean 的数目。
messagecount	计数 统计信息	消息驱动 Bean 收到的消息数。
methodreadycount	范围 统计信息	处于 MethodReady 状态的有状态或无状态会话 Bean 的数目。
passivecount	范围 统计信息	处于 Passive 状态的有状态会话 Bean 的数目。

表 21-2 中列出了可用于 EJB 方法调用的统计信息。

表 21-2 EJB 方法统计信息

属性名称	数据类型	说明
methodstatistic	时间 统计信息	操作被调用的次数；调用期间所花费的总时间等信息。
totalnumerrors	计数 统计信息	方法执行时出现异常的次数。如果为 EJB 容器启用了监视功能，则此统计信息是为无状态和有状态会话 Bean 和实体 Bean 而收集的。
totalnumsuccess	计数 统计信息	方法成功执行的次数。如果为 EJB 容器启用了监视功能，则此统计信息是为无状态和有状态会话 Bean 和实体 Bean 收集的统计信息。
executiontime	计数 统计信息	上次成功/不成功尝试执行方法操作所花费的时间 (ms)。如果在 EJB 容器中启用了监视功能，则此统计信息是为无状态和有状态会话 Bean 和实体 Bean 收集的统计信息。

表 21-3 中列出了 EJB 会话存储的统计信息。

表 21-3 EJB 会话存储统计信息

属性名称	数据类型	说明
currentSize	范围 统计信息	当前位于存储中的钝化会话数或检查点会话数。
activationCount	计数 统计信息	从存储中激活的会话数。
activationSuccessCount	计数 统计信息	从存储中成功激活的会话数。
activationErrorCount	计数 统计信息	上次成功/不成功尝试执行方法操作所花费的时间 (ms)。如果在 EJB 容器中启用了监视功能，则此统计信息是为无状态和有状态会话 Bean 和实体 Bean 收集的统计信息。
passivationCount	计数 统计信息	使用此存储钝化 （取消激活）的会话数。
passivationSuccessCount	计数 统计信息	使用此存储成功钝化的会话数。
passivationErrorCount	计数 统计信息	无法使用此存储钝化的会话数。
expiredSessionCount	计数 统计信息	此存储删除的过期会话数。
passivatedBeanSize	计数 统计信息	由此存储钝化的总字节数，包括总数、最小值和最大值。
passivationTime	计数 统计信息	将 Bean 钝化到存储所花费的时间，包括总时间值、最小值和最大值。
checkpointCount （仅限于 EE）	计数 统计信息	使用此存储进行会话检查点操作的会话数。
checkpointSuccessCount （仅限于 EE）	计数 统计信息	成功进行检查点操作的会话数。
checkpointErrorCount （仅限于 EE）	计数 统计信息	无法进行检查点操作的会话数。
checkpointedBeanSize （仅限于 EE）	值 统计信息	由该存储进行检查点操作的 Bean 的总数。
checkpointTime （仅限于 EE）	时间 统计信息	通过检查点操作将 Bean 放入存储中所花费的时间。

表 21-4 中列出了可用于 EJB 池的统计信息。

表 21-4 EJB 池统计信息

属性名称	数据类型	说明
numbeansinpool	已绑定 范围 统计信息	相关池中的 EJB 数，提供有关池的更改方式的信息。
numthreadswaiting	已绑定 范围 统计信息	等待空闲 Bean 的线程数，指出可能的请求拥塞。
totalbeanscreated	计数 统计信息	自开始收集数据以来相关池中所创建的 Bean 的数目。
totalbeansdestroyed	计数 统计信息	自开始收集数据以来从相关池中销毁的 Bean 的数目。
jmsmaxmessagesload	计数 统计信息	为使消息驱动的 Bean 提供服务而一次加载到 JMS 会话中的最大消息数。默认值为 1。仅适用于消息驱动的 Bean 的池。

表 21-5 中列出了可用于 EJB 高速缓存的统计信息。

表 21-5 EJB 高速缓存统计信息

属性名称	数据类型	说明
cachemisses	已绑定 范围 统计信息	用户请求未在高速缓存中找到 Bean 的次数。
cachehits	已绑定 范围 统计信息	用户请求在高速缓存中找到某条目的次数。
numbeansincache	已绑定 范围 统计信息	高速缓存中 Bean 的数目。这是高速缓存的当前大小。
numpassivations	计数 统计信息	被挂起的数目。仅适用于状态会话 Bean。
numpassivationerrors	计数 统计信息	挂起时的错误数。仅适用于状态会话 Bean。
numexpiredsessionsremoved	计数 统计信息	清除线程删除的过期会话数。仅适用于状态会话 Bean。
numpassivationsuccess	计数 统计信息	成功完成挂起的次数。仅适用于状态会话 Bean。

表 21-6 中列出了可用于计时器的统计信息。

表 21-6 计时器统计信息

统计信息	数据类型	说明
numtimerscreated	计数统计信息	系统中创建的计时器的数目。
numtimersdelivered	计数统计信息	系统所发送的计时器的数目。
numtimersremoved	计数统计信息	从系统中删除的计时器的数目。

Web 容器统计信息

图 21-1 中所示的对象树中包含了 Web 容器。系统为每个单独的 Web 应用程序都显示了 Web 容器统计信息。表 21-7 中显示了可用于 Servlet 的 Web 容器的统计信息，表 21-8 中显示了可用于 Web 模块的统计信息。

表 21-7 Web 容器 (Servlet) 统计信息

统计信息	单位	数据类型	说明
errorcount	次	计数统计信息	响应代码大于或等于 400 的情况的累积次数。
maxtime	毫秒	计数统计信息	Web 容器等待请求的最长时间。
processingtime	毫秒	计数统计信息	处理每个请求所需时间的累积值。处理时间是总请求处理时间除以请求计数所得的平均值。
requestcount	个	计数统计信息	到目前为止所处理的请求总数。

表 21-8 中显示了可用于 Web 模块的统计信息。

表 21-8 Web 容器 （Web 模块）统计信息

统计信息	数据类型	说明
jspcount	计数统计信息	已装入 Web 模块的 JSP 页面的数目。
jspreloadcount	计数统计信息	已重新装入 Web 模块的 JSP 页面的数目。
sessionstotal	计数统计信息	已为 Web 模块创建的会话总数。
activesessionscurrent	计数统计信息	Web 模块的当前处于活动状态的会话数。

表 21-8 Web 容器 （Web 模块）统计信息 (续)

统计信息	数据类型	说明
activesessionshigh	计数统计信息	Web 模块的同时处于活动状态的会话最大数。
rejectedsessionstotal	计数统计信息	Web 模块的被拒绝的会话总数。是指由于允许处于活动状态的会话数已达到最大值而未被创建的会话数。
expiredsessionstotal	计数统计信息	Web 模块的已过期会话的总数。
sessionsize （仅限于 EE）	平均范围统计信息	Web 模块的会话大小。值可以为大、小或平均值，或以字节为单位 （用于序列化会话）。
containerlatency （仅限于 EE）	平均范围统计信息	全部请求等待时间中 Web 容器所占的等待时间。值可以是长、短或平均值。
sessionpersisttime （仅限于 EE）	平均范围统计信息	将 HTTP 会话状态保持到 Web 模块的后端存储中所花费的时间 （以 ms 为单位，短、长或平均值）。
cachedsessionscurrent （仅限于 EE）	计数统计信息	高速缓存在内存中用于 Web 模块的当前会话数。
passivatedsessionscurrent （仅限于 EE）	计数统计信息	用于 Web 模块的已钝化会话的当前数目。

HTTP 服务统计信息

表 21-9 中显示了可用于 HTTP 服务的统计信息。这些统计信息仅适用于平台版。有关企业版上的 HTTP 服务的统计信息，请参见表 21-32。

表 21-9 HTTP 服务统计信息 （仅适用于平台版）

统计信息	单位	数据类型	说明
bytesreceived	字节	计数统计信息	每个请求处理器接收到的字节累积值。
bytessent	字节	计数统计信息	每个请求处理器所发送的字节累积值。
currentthreadcount	个	计数统计信息	当前位于侦听器线程池中的处理线程数。
currentthreadsbusy	个	计数统计信息	处理请求的侦听器线程池中当前正在使用的请求处理线程的数目。

表 21-9 HTTP 服务统计信息 （仅适用于平台版）(续)

统计信息	单位	数据类型	说明
errorcount	个	计数 统计信息	错误计数的累积值，错误计数是指响应代码大于或等于 400 这类情况发生的次数。
maxsparethreads	个	计数 统计信息	可以存在的未使用响应处理线程的最大数目。
minsparethreads	个	计数 统计信息	可以存在的未使用响应处理线程的最小数目。
maxthreads	个	计数 统计信息	侦听器所创建的请求处理线程的最大数目。
maxtime	毫秒	计数 统计信息	处理线程的最长时间。
processing-time	毫秒	计数 统计信息	处理每个请求所花费时间的累积值。处理时间是总请求处理时间除以请求计数所得的平均值。
request-count	个	计数 统计信息	到目前为止所处理的请求总数。

JDBC 连接池统计信息

用于在运行时监视 JDBC 资源，以测量性能并捕获资源使用情况。由于创建 JDBC 连接的成本很高并且常常会导致应用程序出现性能瓶颈问题，因此对 JDBC 连接池释放和创建新连接的方法以及正在等待从特定池中检索连接的线程数的监视是至关重要的。

表 21-10 中显示了可用于 JDBC 连接池的统计信息。

表 21-10 JDBC 连接池统计信息

统计信息	单位	数据类型	说明
numconnfailedvalidation	个	计数 统计信息	从开始时间到上次抽样时间为止在连接池中验证失败的连接总数。
numconnused	个	范围 统计信息	提供连接使用情况统计信息。当前正被使用的连接的总数，以及有关使用过的连接的最大数目 （高水印）的信息。
numconnfree	范围 统计信息	计数 统计信息	上次抽样时池中的空闲连接的总数。

表 21-10 JDBC 连接池统计信息 (续)

统计信息	单位	数据类型	说明
numconntimedout	计数 统计信息	已绑定 范围 统计信息	开始时间与上次抽样时间之间池中的超时连接总数。
averageconnwaittime	时间单位	计数 统计信息	指示尝试与连接器连接池建立连接成功的连接请求的平均等待时间。
waitqueuelength	个	计数 统计信息	队列中正在等待处理的连接请求数。
connectionrequestwaittime		范围 统计信息	连接请求的最长和最短等待时间。当前值表示连接池处理的上一个请求的等待时间。
numconncreated	毫秒	计数统计信息	自上次复位以来创建的物理连接数。
numconndestroyed	个	计数 统计信息	自上次复位以来销毁的物理连接数。
numconnacquired	个	计数 统计信息	从池中获取的逻辑连接数。
numconnreleased	个	计数 统计信息	释放到池中的逻辑连接数。

JMS 连接器服务统计信息

表 21-11 中显示了可用于连接器连接池的统计信息。表 21-12 中显示了连接器工作管理的统计信息。

表 21-11 连接器连接池统计信息

统计信息	单位	数据类型	说明
numconnfailed validation	个	计数 统计信息	从开始时间到上次抽样时间为止在连接池中验证失败的连接总数。
numconnused	个	范围 统计信息	提供连接使用情况统计信息。当前正被使用的连接的总数，以及有关使用过的连接的最大数目的信息 （高水印）。
numconnfree	个	范围 统计信息	上次抽样时池中的空闲连接的总数。
numconntimedout	个	计数 统计信息	开始时间与上次抽样时间之间池中的超时连接总数。

表 21-11 连接器连接池统计信息 (续)

统计信息	单位	数据类型	说明
averageconnwaittime	时间单位	计数 统计信息	连接池处理连接之前这些连接的平均等待时间。
waitqueuelenght	个	计数 统计信息	队列中正在等待处理的连接请求数。
connectionrequest waittime		范围 统计信息	连接请求的最长和最短等待时间。当前值表示连接池处理的上一个请求的等待时间。
numconncreated	毫秒	计数 统计信息	自上次复位以来创建的物理连接数。
numconndestroyed	个	计数 统计信息	自上次复位以来销毁的物理连接数。
numconnacquired	个	计数 统计信息	从池中获取的逻辑连接数。
numconnreleased	个	计数 统计信息	释放到池中的逻辑连接数。

表 21-12 中列出了可用于连接器工作管理的统计信息。

表 21-12 连接器工作管理统计信息

统计信息	数据类型	说明
activeworkcount	范围 统计信息	由连接器执行的工作对象数。
waitqueuelength	范围 统计信息	执行前在队列中等待的工作对象数。
workrequestwaittime	范围 统计信息	工作对象在被执行前所等待的最长和最短时间。
submittedworkcount	计数 统计信息	由连接器模块提交的工作对象数。
rejectedworkcount	计数 统计信息	Application Server 拒绝的工作对象数。
completedworkcount	计数 统计信息	完成的工作对象数。

ORB 中连接管理器的统计信息

表 21-13 中列出了可用于 ORB 中连接管理器的统计信息。

表 21-13 ORB 中连接管理器的统计信息

统计信息	单位	数据类型	说明
connectionsidle	个	计数统计信息	提供与 ORB 的空闲连接的总数。
connectionsinuse	个	计数统计信息	提供与 ORB 的正在使用的连接总数。
totalconnections	个	绑定范围统计信息	与 ORB 的连接总数。

线程池统计信息

表 21-14 中显示了可用于线程池的统计信息。

表 21-14 线程池统计信息

统计信息	单位	数据类型	说明
averagetimeinqueue	毫秒	范围统计信息	在被处理之前请求在队列中等待的平均时间 （以毫秒为单位）。
averageworkcompletion-time	毫秒	范围统计信息	完成分配所花费的平均时间 （以毫秒为单位）。
currentnumberofthreads	个	绑定范围统计信息	当前的请求处理线程数。
numberofavailablethreads	个	计数统计信息	可用的线程数。
numberofbusythreads	个	计数统计信息	处于忙碌状态的线程数。
totalworkitemsadded	个	计数统计信息	到目前为止添加到工作队列中的工作项目总数。

事务服务统计信息

事务服务允许客户机冻结事务子系统，以回滚事务并确定冻结时正在进行的事务。表 21-15 中显示了可用于事务服务的统计信息。

表 21-15 事务服务统计信息

统计信息	数据类型	说明
activecount	计数统计信息	当前处于活动状态的事务数。
activeids	字符串 统计信息	当前处于活动状态的事务的 ID。冻结事务服务后，可以回滚所有此类事务。
committedcount	计数统计信息	已提交的事务数。
rolledbackcount	计数统计信息	已回滚的事务数。
state	字符串 统计信息	表示事务是否已被冻结。

Java 虚拟机 (JVM) 统计信息

JVM 具有始终处于启用状态的可监视属性。表 21-16 中显示了可用于 JVM 的统计信息。

表 21-16 JVM 统计信息

统计信息	数据类型	说明
heapsize	绑定范围 统计信息	具有 JVM 的内存堆大小的较高和较低绑定的驻留内存轨迹。
uptime	计数统计信息	JVM 已运行的时间。

J2SE 5.0 中的 *JVM* 统计信息

如果 Application Server 被配置为运行 J2SE 5.0 版或更高版本，则可以从 JVM 中获得附加监视信息。将监视级别设置为 “低”以启用这些附加信息的显示。将监视级别设置为 “高”还可以查看与系统中每个活动线程相关的信息。有关 J2SE 5.0 中可用的附加监视功能的更多信息，请参见以下 URL 中标题为 《Monitoring and Management for the Java Platform》的文档：

http://java.sun.com/j2se/1.5.0/docs/guide/management/

在以下位置讨论了 J2SE 5.0 监视工具：

http://java.sun.com/j2se/1.5.0/docs/tooldocs/#manage

表 21-17 中显示了可用于在 J2SE 5.0 的 JVM 中进行类装入的统计信息。

表 21-17 J2SE 5.0 的 JVM 统计信息－类装入

统计信息	数据类型	说明
loadedclasscount	计数统计信息	当前装入 JVM 的类的数目。
totalloadedclasscount	计数统计信息	自 JVM 开始执行以来已装入的类的总数。
unloadedclasscount	计数统计信息	自 JVM 开始执行以来已从 JVM 中卸载的类的数目。

表 21-18 中显示了可用于在 J2SE 5.0 的 JVM 中进行编译的统计信息。

表 21-18 J2SE 5.0 的 JVM 统计信息－编译

统计信息	数据类型	说明
totalcompilationtime	计数统计信息	编译所花费的累积时间 （以毫秒为单位）。

表 21-19 中显示了可用于在 J2SE 5.0 的 JVM 中进行垃圾收集的统计信息。

表 21-19 J2SE 5.0 的 JVM 统计信息－垃圾收集

统计信息	数据类型	说明
collectioncount	计数统计信息	已发生的收集的总数。
collectiontime	计数统计信息	累积的收集时间 （以毫秒为单位）。

表 21-20 中显示了可用于 J2SE 5.0 的 JVM 中内存的统计信息。

表 21-20 J2SE 5.0 的 JVM 统计信息－内存

统计信息	数据类型	说明
objectpendingfinalizationcount	计数统计信息	暂挂结束操作的对象的大约数目。
initheapsize	计数统计信息	最初由 JVM 请求的堆的大小。
usedheapsize	计数统计信息	当前正在使用的堆的大小。
maxheapsize	计数统计信息	可用于内存管理的最大内存容量（以字节为单位）。

表 21-20 J2SE 5.0 的 JVM 统计信息－内存 (续)

统计信息	数据类型	说明
committedheapsize	计数统计信息	确认可由 JVM 使用的内存容量 （以字节为单位）。
initnonheapsize	计数统计信息	最初由 JVM 请求的非堆区域的大小。
usednonheapsize	计数统计信息	当前正在使用的非堆区域的大小。
maxnonheapsize	计数统计信息	可用于内存管理的最大内存容量（以字节为单位）。
committednonheapsize	计数统计信息	确认可由 JVM 使用的内存容量 （以字节为单位）。

表 21-21 中显示了可用于 J2SE 5.0 的 JVM 中操作系统的统计信息。

表 21-21 J2SE 5.0 的 JVM 统计信息－操作系统

统计信息	数据类型	说明
arch	字符串统计信息	操作系统体系结构。
availableprocessors	计数统计信息	可用于 JVM 的处理器的数目。
name	字符串统计信息	操作系统名称。
version	字符串统计信息	操作系统版本。

表 21-22 中显示了可用于 J2SE 5.0 的 JVM 中运行的统计信息。

表 21-22 J2SE 5.0 的 JVM 统计信息－运行

统计信息	数据类型	说明
name	字符串统计信息	代表正在运行的 JVM 的名称
vmname	字符串统计信息	JVM 实现名称。
vmvendor	字符串统计信息	JVM 实现供应商。
vmversion	字符串统计信息	JVM 实现版本。
specname	字符串统计信息	JVM 规范名称。
specvendor	字符串统计信息	JVM 规范供应商。
specversion	字符串统计信息	JVM 规范版本。
managementspecversion	字符串统计信息	由 JVM 实现的管理规范版本。

表 21-22 J2SE 5.0 的 JVM 统计信息—运行 (续)

统计信息	数据类型	说明
classpath	字符串统计信息	系统类加载器搜索类文件时所使用的类路径。
librarypath	字符串统计信息	Java 库路径。
bootclasspath	字符串统计信息	引导类加载器搜索类文件时所使用的类路径。
inputarguments	字符串统计信息	传递给 JVM 的输入参数。不包括 main 方法的参数。
uptime	计数统计信息	JVM 的正常运行时间 （以毫秒为单位）。

表 21-23 中显示了可用于 J2SE 5.0 的 JVM 中 ThreadInfo 的统计信息。

表 21-23 J2SE 5.0 的 JVM 统计信息—线程信息

统计信息	数据类型	说明
threadid	计数统计信息	线程 ID。
threadname	字符串统计信息	线程名称。
threadstate	字符串统计信息	线程状态。
blockedtime	计数统计信息	线程进入 "BLOCKED" 状态以来所经历的时间 （以毫秒为单位）。如果已禁用线程争用监视，则返回 -1。
blockedcount	计数统计信息	线程进入 BLOCKED 状态的总次数。
waitedtime	计数统计信息	线程处于 "WAITING" 状态所经历的时间 （以毫秒为单位）。如果已禁用线程争用监视，则返回 -1。
waitedcount	计数统计信息	线程处于 WAITING 或 TIMED_WAITING 状态的总次数。
lockname	字符串统计信息	监视锁的字符串表示，该监视锁为线程被阻塞而无法进入或者线程等待通过 Object.wait 方法接收通知。
lockownerid	计数统计信息	保存某个对象的监视锁的线程 ID，该线程在该对象上发生阻塞。
lockownername	字符串统计信息	保存某个对象的监视锁的线程名称，该线程在该对象上发生阻塞。
stacktrace	字符串统计信息	与该线程相关的堆栈追踪。

表 21-24 中显示了可用于 J2SE 5.0 的 JVM 中线程的统计信息。

表 21-24 J2SE 5.0 的 JVM 统计信息－线程

统计信息	数据类型	说明
threadcount	计数统计信息	当前的活动守护线程和非守护线程数。
peakthreadcount	计数统计信息	自 JVM 启动或峰复位以来的峰活动线程计数。
totalstartedthreadcount	计数统计信息	自 JVM 启动以来创建和/或启动的线程总数。
daemonthreadcount	计数统计信息	当前的活动守护线程数。
allthreadids	字符串统计信息	所有活动线程 ID 列表。
currentthreadcputime	计数统计信息	如果已启用 CPU 时间测量，则表示当前线程的 CPU 时间 （以纳秒为单位）。如果已禁用 CPU 时间测量，则返回 -1。
monitordeadlockedthreads	字符串统计信息	处于监视死锁状态的线程 ID 列表。

生产 Web 容器 (PWC) 统计信息

可用于 Application Server 的企业版 (EE) 上的以下 PWC 组件和服务的统计信息：

- 表 21-25， PWC 虚拟服务器
- 表 21-26， PWC 请求
- 表 21-27， PWC 文件高速缓存
- 表 21-28， PWC 保持活动
- 表 21-29， PWC DNS
- 表 21-30， PWC 线程池
- 表 21-31， PWC 连接队列
- 表 21-32， PWC HTTP 服务

表 21-25 中列出了 PWC 虚拟服务器的统计信息。

表 21-25 PWC 虚拟服务器统计信息 （仅限于 EE）

属性名称	数据类型	说明
id	字符串 统计信息	虚拟服务器的 ID。
mode	字符串 统计信息	虚拟服务器所处的模式。选项包括 unknown 或 active。
hosts	字符串 统计信息	由该虚拟服务器提供服务的主机的名称。
interfaces	字符串 统计信息	配置了虚拟服务器的接口 （侦听器）的类型。

表 21-26 中列出了可用于 PWC 请求的统计信息。

表 21-26 PWC 请求统计信息 （仅限于 EE）

属性名称	数据类型	说明
method	字符串 统计信息	用于请求的方法。
uri	字符串 统计信息	处理的上一个 URI。
countrequests	计数 统计信息	已处理的请求数。
countbytestransmitted	计数 统计信息	传输的字节数；如果此信息不可用，则值为 0。
countbytesreceived	计数 统计信息	收到的字节数；如果此信息不可用，则值为 0。
ratebytesreceived	计数 统计信息	在某段服务器定义的时间间隔内数据的传输速率；如果此信息不可用，则值为 0。
maxbytestransmissionrate	计数 统计信息	在某段服务器定义的时间间隔内数据的最大传输速率；如果此信息不可用，则值为 0。
countopenconnections	计数 统计信息	当前打开的连接数；如果此信息不可用，则值为 0。
maxopenconnections	计数 统计信息	同时打开的连接的最大数目；如果此信息不可用，则值为 0。
count2xx	计数 统计信息	代码为 2XX 的响应的总数。

表 21-26 PWC 请求统计信息 （仅限于 EE） (续)

属性名称	数据类型	说明
count3xx	计数 统计信息	代码为 3XX 的响应的总数。
count4xx	计数 统计信息	代码为 4XX 的响应的总数。
count5xx	计数 统计信息	代码为 5XX 的响应的总数。
countother	计数 统计信息	具有其他响应代码的响应总数。
count200	计数 统计信息	代码为 200 的响应的总数。
count302	计数 统计信息	代码为 302 的响应的总数。
count304	计数 统计信息	代码为 304 的响应的总数。
count400	计数 统计信息	代码为 400 的响应的总数。
count401	计数 统计信息	代码为 401 的响应的总数。
count403	计数 统计信息	代码为 403 的响应的总数。
count404	计数 统计信息	代码为 404 的响应的总数。
count503	计数 统计信息	代码为 503 的响应的总数。

高速缓存信息部分提供了有关文件高速缓存当前的使用方式的信息。表 21-27 中列出了 PWC 文件高速缓存的统计信息。

表 21-27 PWC 文件高速缓存统计信息 （仅限于 EE）

属性名称	数据类型	说明
flagenabled	计数 统计信息	指示是否启用了文件高速缓存。禁用时有效值为 0 ；启用时有效值为 1。
secondsmaxage	计数 统计信息	有效高速缓存条目的最长生存期 （以秒为单位）。

表 21-27 PWC 文件高速缓存统计信息 （仅限于 EE）(续)

属性名称	数据类型	说明
countentries	计数 统计信息	当前的高速缓存条目数。一个高速缓存条目代表一个 URI。
maxentries	计数 统计信息	并发高速缓存条目的最大数目。
countopenentries	计数 统计信息	与打开的文件关联的条目数。
maxopenentries	计数 统计信息	与打开的文件关联的并发高速缓存条目的最大数目。
sizeheapcache	计数 统计信息	用于高速缓存内容的堆空间。
maxheapcachesize	计数 统计信息	用于高速缓存文件内容的最大堆空间。
sizemmapcache	计数 统计信息	内存映射的文件内容所使用的地址空间大小。
maxmmapcachesize	计数 统计信息	文件高速缓存用于内存映射的文件内容的最大地址空间大小。
counthits	计数 统计信息	高速缓存查找成功的次数。
countmisses	计数 统计信息	高速缓存查找失败的次数。
countinfohits	计数 统计信息	文件信息查找成功的次数。
countinfomisses	计数 统计信息	高速缓存的文件信息丢失的数目。
countcontenthits	计数 统计信息	高速缓存的文件内容的命中次数。
countcontentmisses	计数 统计信息	文件信息查找失败的次数。

本节提供了有关服务器的 HTTP 级保持活动的系统的信息。表 21-28 中列出了可用于 PWC 保持活动的统计信息。

表 21-28 PWC 保持活动统计信息 （仅限于 EE）

属性名称	数据类型	说明
countconnections	计数 统计信息	处于保持活动模式的连接数。
maxconnections	计数 统计信息	允许同时处于保持活动模式的最大连接数。
counthits	计数 统计信息	处于保持活动模式的连接随后进行了有效请求的总次数。
countflushes	计数 统计信息	服务器关闭保持活动的连接的次数。
countrefusals	计数 统计信息	服务器可能由于有太多的持久性连接而无法将连接传递到保持活动线程的次数。
counttimeouts	计数 统计信息	服务器因客户机连接超时且没有任何活动而终止保持活动连接的次数。
secondstimeout	计数 统计信息	关闭空闲保持活动连接之前经历的时间 （以秒为单位）。

DNS 高速缓存高速缓存 IP 地址和 DNS 名称。默认情况下，服务器的 DNS 高速缓存处于禁用状态。一个高速缓存条目代表一个 IP 地址或 DNS 名称查找。表 21-29 中列出了可用于 PWC DNS 的统计信息。

表 21-29 PWC DNS 统计信息 （仅限于 EE）

属性名称	数据类型	说明
flagcacheenabled	计数 统计信息	指示是否启用了 DNS 高速缓存。禁用时为 0 ； 启用时为 1。
countcacheentries	计数 统计信息	当前位于高速缓存中的 DNS 条目数。
maxcacheentries	计数 统计信息	高速缓存中可容纳的 DNS 条目的最大数目。
countcachehits	计数 统计信息	DNS 高速缓存查找成功的次数。
countcachemisses	计数 统计信息	DNS 高速缓存查找失败的次数。

表 21-29 PWC DNS 统计信息 （仅限于 EE）(续)

属性名称	数据类型	说明
flagasyncenabled	计数 统计信息	指示是否启用了异步 DNS 查找。禁用时为 0 ； 启用时为 1。
countasyncnamelookups	计数 统计信息	异步 DNS 名称查找的总数。
countasyncaddrlookups	计数 统计信息	异步 DNS 地址查找的总数。
countasynclookupsinprogress	计数 统计信息	正在进行的异步查找的数目。

表 21-30 中列出了 PWC 线程池的统计信息。

表 21-30 PWC 线程池统计信息 （仅限于 EE）

属性名称	数据类型	说明
id	字符串 统计信息	线程池 ID。
countthreadsidle	计数 统计信息	当前处于空闲状态的请求处理线程数。
countthreads	计数 统计信息	当前的请求处理线程的数目。
maxthreads	计数 统计信息	可同时存在的请求处理线程的最大数目。
countqueued	计数 统计信息	排队等候此线程池处理的请求数。
peakqueued	计数 统计信息	队列中同时容纳的最大请求数。
maxqueued	计数 统计信息	队列一次可容纳的最大请求数。

连接队列是指请求被处理前容纳这些请求的队列。连接队列的统计信息显示队列中的会话数以及连接被接受前的平均延迟时间。表 21-31 中列出了 PWC 连接队列的统计信息。

表 21-31 PWC 连接队列统计信息 （仅限于 EE）

属性名称	数据类型	说明
id	字符串 统计信息	连接队列的 ID。
counttotalconnections	计数 统计信息	已接受的连接总数。
countqueued	计数 统计信息	当前位于队列中的连接数。
peakqueued	计数 统计信息	队列中同时容纳的最大连接数。
maxqueued	计数 统计信息	连接队列的最大大小。
countoverflows	计数 统计信息	队列太满而无法容纳更多连接的次数。
counttotalqueued	计数 统计信息	排队等候的连接总数。某个给定连接可能被多次排队，因此 counttotalqueued 可能大于或等于 counttotalconnections。
tickstotalqueued	计数 统计信息	连接在队列中所花费的周期总数。周期是由系统决定的时间单位。
countqueued1minuteaverage	计数 统计信息	前 1 分钟内处于排队状态的平均连接数。
countqueued5minuteaverage	计数 统计信息	前 5 分钟内处于排队状态的平均连接数。
countqueued15minuteaverage	计数 统计信息	前 15 分钟内处于排队状态的平均连接数。

表 21-32 中列出了 PWC HTTP 服务的统计信息。

表 21-32 PWC HTTP 服务统计信息 （仅限于 EE）

属性名称	数据类型	说明
id	字符串 统计信息	HTTP 服务的实例名称。
versionserver	字符串 统计信息	HTTP 服务的版本号。
timestarted	字符串 统计信息	启动 HTTP 服务的时间 (GMT)。
secondsrunning	计数 统计信息	HTTP 服务启动以来所经历的时间 （以秒为单位）。
maxthreads	计数 统计信息	每个实例中的最大工作线程数。
maxvirtualservers	计数 统计信息	每个实例中可以配置的最大虚拟服务器数目。
flagprofilingenabled	计数 统计信息	是否启用了 HTTP 服务性能探查。有效值为 0 或 1。
flagvirtualserveroverflow	计数 统计信息	指示是否配置了超过 maxvirtualservers 的虚拟服务器。如果此属性设置为 1，则无法对所有虚拟服务器的统计信息进行跟踪。
load1minuteaverage	计数 统计信息	前 1 分钟内请求的平均负载。
load5minuteaverage	计数 统计信息	前 5 分钟内请求的平均负载。
load15minuteaverage	计数 统计信息	前 15 分钟内请求的平均负载。
ratebytestransmitted	计数 统计信息	在某个服务器定义的时间间隔内数据的传输速率。如果此信息不可用，则结果为 0。
ratebytesreceived	计数 统计信息	在某个服务器定义的时间间隔内数据的接收速率。如果此信息不可用，则结果为 0。

有关启用和禁用监视功能的管理控制台任务

使用管理控制台配置监视级别

1. 访问 “监视服务”页面。要完成此操作，请执行以下步骤：

   a. 在树中，展开 “配置”节点。

   b. 在树中，展开要配置为用于监视的服务器实例的节点，例如 `server-config`。

   c. 在树中，选择 “监视”。

2. 在 “监视服务”页面中，从要更改其监视级别的组件或服务对应的组合框中选择适当的值。

   默认情况下，除了始终可监视的 Java 虚拟机 (JVM) 以外，对于其他所有组件和服务的监视功能都是禁用的。要启用监视功能，请在组合框中选择 “低”或 “高”。要禁用监视功能，请在组合框中选择 “关闭”。可以启用或禁用以下组件和服务的监视功能：

   - **JVM** －将此选项的监视级别设置为 “低”以监视 Java 虚拟机。
   - **HTTP 服务**－将此选项的监视级别设置为 “低”以监视所有 HTTP 侦听器和虚拟服务器。
   - **事务服务**－将此选项的监视级别设置为 “低”以监视任一事务子系统。
   - **JMS/ 连接器服务**－将此选项的监视级别设置为 “低”以监视任一 Java 消息服务 (JMS)。
   - **ORB** －将此选项的监视级别设置为 “低”以监视由 Application Server 内核及其连接管理器使用的系统 ORB。
   - **Web 容器**－将此选项的监视级别设置为 “低”以监视所有已部署的 Servlet。
   - **EJB 容器**－将此选项的监视级别设置为 “低”以监视所有已部署的 EJB、EJB 池和 EJB 高速缓存。将此方法设置为 “高”还可以监视 EJB 商业方法。
   - **JDBC 连接池**－将此选项的监视级别设置为 “低”以监视所有 JDBC 连接池。
   - **线程池**－将此选项的监视级别设置为 “低”以监视所有线程池。

3. 单击“保存”。

此版本中没有“其他监视服务属性”，因此请忽略“其他属性”表。

等效的 asadmin 命令为：set，例如，要启用对 HTTP 服务的监视功能，请使用以下 asadmin 命令：

```
asadmin> set --user admin_user
server.monitoring-service.module-monitoring-levels.http-service=LOW
```

使用 asadmin 工具配置监视

要关闭监视，或设置监视组件或服务的级别，您可以按照 “使用管理控制台配置监视级别”中的说明使用管理控制台，也可以按照本节中的说明使用 asadmin 工具。

1. 使用 get 命令可以查找哪些服务和组件当前已启用监视功能：

   ```
   asadmin> get --user admin_user
   server.monitoring-service.module-monitoring-levels.*
   ```

   返回：

   ```
   server.monitoring-service.module-monitoring-levels.
   connector-connection-pool = OFF
   server.monitoring-service.module-monitoring-levels.
   connector-service = OFF
   server.monitoring-service.module-monitoring-levels.ejb-container = OFF
   server.monitoring-service.module-monitoring-levels.http-service = OFF
   server.monitoring-service.module-monitoring-levels.jdbc-connection-pool
   = OFF
   server.monitoring-service.module-monitoring-levels.jms-service = OFF
   server.monitoring-service.module-monitoring-levels.jvm = OFF
   server.monitoring-service.module-monitoring-levels.orb = OFF
   server.monitoring-service.module-monitoring-levels.thread-pool = OFF
   server.monitoring-service.module-monitoring-levels.transaction-service
   = OFF
   server.monitoring-service.module-monitoring-levels.web-container = OFF
   ```

2. 使用 set 命令可以启用监视功能。

   例如，要启用对 HTTP 服务的监视，请输入以下命令：

   ```
   asadmin> set --user admin_user
   server.monitoring-service.module-monitoring-levels.http-service=LOW
   ```

   要禁用监视，请使用 set 命令并将监视级别指定为 OFF。

有关查看监视数据的管理控制台任务

在管理控制台中查看监视数据

要使用 Application Server 管理控制台查看服务器实例中部署的组件或服务的监视数据，请执行以下步骤。有关每个组件或服务的属性的更多说明，请参阅“关于受监视的组件和服务的统计信息”。

1. 访问 “监视”页面。要完成此操作，请执行以下步骤：
   a. 在树组件中，展开 “独立实例”节点，例如 `server (Admin Server)`。
   b. 从列表中选择一个独立服务器实例。
   c. 选择 “监视”页面。
   d. 在 “监视”页面上选择 “监视”选项卡。
2. 在 “查看”列表中，选择已部署到服务器实例并已为其启用了监视功能的组件或服务。

   选定组件或服务的监视数据显示在 “查看”字段下。有关可监视属性的说明，请参阅 “关于受监视的组件和服务的统计信息”。

   如果已启用 JVM、服务器、线程池、 HTTP 服务和事务服务的监视功能，则可以在此页面中查看这些组件和服务的监视数据。“关于可监视对象的树结构”中显示了说明这些组件和服务的组织方式的图表。
3. 如果在此列表中没有看到要监视的组件或服务，请选择 “配置监视”链接来启用和禁用选定组件和服务的监视功能。选择 “关闭”以禁用对组件或服务的监视。选择 “低”或 “高”以启用对组件或服务的监视。

   有关启用和禁用监视功能的更多信息，请参阅 “使用管理控制台配置监视级别”或 “使用 asadmin 工具配置监视”。

4. 选择 “监视”页面的 “**应用程序**”选项卡可以查看已部署到服务器实例并已为其启用了监视功能的应用程序组件的监视数据。从 “应用程序”列表中选择应用程序。从 “组件”列表中选择特定组件。

   如果未显示应用程序或组件的监视数据，请选择 “配置监视”链接来启用或禁用组件或服务的监视功能。要监视应用程序，请打开执行这些应用程序的容器：例如，对于 Web 应用程序的 Web 容器和 / 或 EJB 应用程序的 EJB 容器，请选择 “低”或 “高”。

   如果未显示应用程序的监视数据，则该应用程序很可能不存在或未运行。仅当应用程序存在、已启用对该程序的监视功能且应用程序正在运行时，应用程序监视数据才可用。一旦执行应用程序，则将在监视注册表登记该应用程序并显示其监视数据。

   使用管理控制台监视远程应用程序和实例。远程实例必须正在运行并已设置配置才能执行此操作。

   选定组件的监视数据显示在选定组件下面。有关可监视属性的说明，请参阅 “关于受监视的组件和服务的统计信息”。可以在 “关于可监视对象的树结构”中查看说明这些组件和服务如何针对应用程序进行组织的图表。

5. 选择 “**资源**”页面可以查看资源的监视数据，这些资源已被部署到服务器实例中并且已启用对这些资源的监视功能。从 “查看”列表中选择资源。如果未显示您要查看其监视数据的资源，请选择 “配置监视”链接来启用或禁用资源的监视功能。

   如果未显示资源的监视数据，则该资源很可能不存在或未运行。仅当资源存在、已在 HIGH 级别启用该资源的监视功能并且该资源正在运行的情况下，资源监视数据才可用。例如，如果您创建了一个 JDBC 连接器服务，但使用该连接器服务的应用程序尚未从服务请求连接器，则尚未创建该服务。因而，不存在任何服务，也没有任何可用的监视数据。一旦执行 JDBC 应用程序且该应用程序从服务请求连接器，则将在监视注册表中注册该应用程序并显示其监视数据。

   选定组件或服务的监视数据显示在 “查看”字段下。有关可监视属性的说明，请参阅 “关于受监视的组件和服务的统计信息”。可以在 “关于可监视对象的树结构”中查看说明这些组件和服务如何针对资源进行组织的图表。

6. 选择 “**事务**”页面以冻结事务子系统，进而回滚事务并确定冻结时正在进行的事务。要启用事务服务的监视功能，请选择 “配置监视”链接并确保将 “事务服务”设置为 LOW。要冻结事务服务以回滚事务，请选择 “冻结”。要回滚事务，请选中事务旁边的复选框并单击 “回滚”。

等效的 `asadmin` 命令为：`get --monitor`，例如，要查看 JVM 的监视数据，请使用以下 `asadmin` 命令：

```
asadmin> get --monitor server.jvm.*
```

使用 asadmin 工具查看监视数据

使用 asadmin 工具查看监视数据

要使用 asadmin 工具查看监视数据，请使用后跟可监视对象的点式名称的 asadmin list 和 asadmin get 命令。使用 asadmin 工具查看监视数据的常规步骤如下：

1. 要查看可监视对象的名称，请使用 asadmin list 命令。例如，要查看应用程序组件和子系统（已启用对服务器实例上的这些组件和子系统的监视功能）的列表，请在终端窗口中键入以下命令：

   ```
   asadmin> list --monitor server
   ```

   上述命令将返回已启用监视功能的应用程序组件和子系统的列表，例如：

   ```
   server.resources
   server.connector-service
   server.orb
   server.jms-service
   server.jvm
   server.applications
   server.http-service
   server.thread-pools
   ```

   有关使用 list 命令的详细示例，请参阅 “list 和 get 命令的示例”。有关可与 list 命令一起使用的带点的名称的详细信息，请参阅 “了解和指定带点的名称”。

2. 要显示已启用监视功能的应用程序组件或子系统的监视统计信息，请使用 asadmin get 命令。要获得统计信息，请在终端窗口中键入 asadmin get 命令，并指定在先前步骤中由 list 命令显示的名称。以下示例尝试获取某个特定对象的子系统的所有属性：

   ```
   asadmin> get --monitor server.jvm.*
   ```

此命令返回以下属性和数据：

```
server.jvm.dotted-name = server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information
about the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

Sun Java System Application Server Enterprise Edition 8.1 2005Q1 有关使用 `get` 命令的详细示例，请参阅 “list 和 get 命令的示例”。有关可与 `get` 命令一起使用的带点的名称的详细信息，请参阅 “了解和指定带点的名称”。

了解和指定带点的名称

在 `asadmin list` 和 `get` 命令中，指定可监视对象的带点的名称。所有子对象都是使用点 (.) 字符作为分隔符来指定的，因而这些子对象的名称也称为**带点的名称**。如果子节点是单元素类型，则只需要使用监视对象类型来指定对象；否则，需要使用 `type.name` 名称格式来指定对象。

例如， `http-service` 就是其中一种有效的可监视对象类型，并且是单元素类型。要指定表示实例 `server` 的 `http-service` 的单元素类型子节点，则带点的名称为：

```
server.http-service
```

再比如， `application` 是一种有效的可监视对象类型，并且是非单元素类型。例如，要指定表示应用程序 Petstore 的非单元素类型子节点，则带点的名称为：

```
server.applications.petstore
```

带点的名称还可以指定可监视对象中的特定属性。例如，`http-service` 具有一个名为 `bytesreceived-lastsampletime` 的可监视属性。以下名称指定了 `bytesreceived` 属性：

```
server.http-service.server.http-listener-1.
  bytesreceived-lastsampletime
```

管理员不需要知道 `asadmin list` 和 `get` 命令的有效带点名称。使用 `list` 命令可以显示可用的可监视对象，而使用带有通配符参数的 `get` 命令可以检查任意可监视对象的所有可用属性。

使用具有带点名称的 `list` 和 `get` 命令的基本假设为：

- 使用任何具有带点名称且后面**不**跟通配符 (`*`) 的 `list` 命令，得到的结果为当前节点的直接子节点。例如，`list --monitor server` 列出了属于 `server` 节点的所有直接子节点。
- 使用任何具有带点名称且后面跟有 `.*` 形式的通配符的 `list` 命令，得到的结果为当前节点的子节点分层树。例如，`list --monitor server.applications.*` 列出了 `applications` 的所有子节点及其后续子节点等。
- 使用任何具有带点名称并且前面或后面带有 **dottedname* 或 *dotted * name* 或 **`dotted name *`** 形式的通配符的 `list` 命令，得到的结果为所有节点以及符合提供的匹配模式所创建的常规表现的子节点。
- 使用后跟 `.*` 或 `*` 的 `get` 命令，得到的结果为属于要匹配的当前节点的属性集及其值。

有关更多信息，请参阅 “list 和 get 命令在所有级别上的预期输出”。

list 和 get 命令的示例

本节包括以下主题：

- list --monitor 命令的示例
- get --monitor 命令的示例
- Petstore 示例

list --monitor 命令的示例

`list` 命令提供有关当前监视的指定服务器实例名称的应用程序组件和子系统的信息。使用此命令，您可以查看某个服务器实例的可监视组件和子组件。有关 `list` 示例的更完整的列表，请参见 “list 和 get 命令在所有级别上的预期输出”。

示例 *1*

```
asadmin> list --monitor server
```

上述命令将返回已启用监视功能的应用程序组件和子系统的列表，例如：

```
server.resources
server.orb
server.jvm
server.jms-service
server.connector-service
server.applications
server.http-service
server.thread-pools
```

还可以列出指定的服务器实例中当前所监视的应用程序。当使用 get 命令从某个应用程序中获取特定的监视统计信息时，这会很有用。

示例 *2*

```
asadmin> list --monitor server.applications
```

返回：

```
server.applications.adminapp
server.applications.admingui
server.applications.myApp
```

有关更复杂的示例，请参见 “Petstore 示例”。

get --monitor 命令的示例

此命令检索以下监视信息：

- 组件或子系统中监视的所有属性
- 组件或子系统中监视的特定属性

如果特定组件或子系统中不存在所请求的属性，将返回一个错误。类似地，如果组件或子系统中所请求的特定属性处于非活动状态，也会返回一个错误。

有关使用 get 命令的详细信息，请参见 “list 和 get 命令在所有级别上的预期输出”。

示例 *1*

尝试从某个子系统中获取某个特定对象的所有属性：

```
asadmin> get --monitor server.jvm.*
```

返回：

```
server.jvm.dotted-name = server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information
about the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

示例 *2*

尝试获取某个 J2EE 应用程序的所有属性：

```
asadmin> get --monitor server.applications.myJ2eeApp.*
```

返回：

```
No matches resulted from the wildcard expression.
CLI137 Command get failed.
```

该 J2EE 应用程序级别上没有暴露可监视的属性，因而显示此回复。

示例 *3*

尝试获取某个子系统的某个特定属性：

```
asadmin> get --monitor server.jvm.uptime-lastsampletime
```

返回：

```
server.jvm.uptime-lastsampletime = 1093215374813
```

示例 *4*

尝试获取某个子系统属性中的某个未知属性：

```
asadmin> get --monitor server.jvm.badname
```

返回：

```
No such attribute found from reflecting the corresponding Stats
interface: [badname]
CLI137 Command get failed.
```

Petstore 示例

下例说明了如何将 asadmin 工具用于监视目的。

用户要检查当将样例 Petstore 应用程序部署到 Application Server 后在该应用程序中调用某个方法的次数。部署该应用程序的实例名称为 server。这里同时使用了 list 和 get 命令，以访问针对方法的所需统计信息。

1. 启动 Application Server 和 asadmin 工具。
2. 设置一些有用的环境变量，以避免使用每个命令时都输入这些变量：

   ```
   asadmin>export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=admin123
   ```

   ```
   asadmin>export AS_ADMIN_HOST=localhost AS_ADMIN_PORT=4848
   ```

3. 列出实例 server 的可监视组件：

   ```
   asadmin>list --monitor server*
   ```

   返回 （输出类似于以下内容）：

   ```
   server
   server.applications
   server.applications.CometEJB
   server.applications.ConverterApp
   server.applications.petstore
   server.http-service
   server.resources
   server.thread-pools
   ```

   可监视组件列表包括 thread-pools、 http-service、 resources 以及所有已部署（并已启用）的 applications。

4. 列出 Petstore 应用程序中的可监视子组件 （可以用 -m 代替 --monitor）：

   ```
   asadmin>list -m server.applications.petstore
   ```

   返回：

   ```
   server.applications.petstore.signon-ejb_jar
   server.applications.petstore.catalog-ejb_jar
   server.applications.petstore.uidgen-ejb_jar
   server.applications.petstore.customer-ejb_jar
   server.applications.petstore.petstore-ejb_jar
   server.applications.petstore.petstore\.war
   server.applications.petstore.AsyncSenderJAR_jar
   server.applications.petstore.cart-ejb_jar
   ```

5. 列出 Petstore 应用程序的 EJB 模块 signon-ejb_jar 中的可监视子组件：

   ```
   asadmin>list -m server.applications.petstore.signon-ejb_jar
   ```

   返回：

   ```
   server.applications.petstore.signon-ejb_jar.SignOnEJB
   server.applications.petstore.signon-ejb_jar.UserEJB
   ```

6. 列出 Petstore 应用程序的 EJB 模块 signon-ejb_jar 的实体 Bean UserEJB 中的可监视子组件：

   ```
   asadmin>list -m server.applications.petstore.signon-ejb_jar.UserEJB
   ```

   返回 （出于空间考虑，删除了带点的名称）：

   ```
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-cache
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-pool
   ```

7. 列出实体 Bean UserEJB （位于 Petstore 应用程序的 EJB 模块 signon-ejb_jar 中）的方法 getUserName 的可监视子组件：

   ```
   asadmin>list  -m server.applications.petstore.signon-ejb_jar.
   UserEJB.bean-methods.getUserName
   ```

   返回：

   ```
   Nothing to list at server.applications.petstore.signon-ejb_jar.
   UserEJB.bean-methods.getUserName. To get the valid names beginning with
   a string, use the wildcard "*" character.For example, to list all names
   that begin with "server", use ìlist server*î.
   ```

8. 该方法没有可监视的子组件。获取方法 getUserName 的所有可监视统计信息。

```
asadmin>get -m server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName.*
```

返回：

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-description = Provides the time in
milliseconds spent during the last successful/unsuccessful attempt
to execute the operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-lastsampletime = 1079981809259
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-description = Provides the number of times
an operation was called, the total time that was spent during the
invocation and so on.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-lastsampletime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-maxtime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-mintime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-totaltime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-unit =
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-description = Provides the total number of
errors that occured during invocation or execution of an operation.
```

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-lastsampletime = 1079981809273
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-name = TotalNumErrors
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-description = Provides the total number of
successful invocations of the method.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-lastsampletime = 1079981809255
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-name = TotalNumSuccess
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-unit = count
```

9. 如果还需要获取执行时间等特定统计信息，请使用如下命令：

```
asadmin>get -m server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName.executiontime-count
```

返回：

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 1
```

list 和 get 命令在所有级别上的预期输出

下表显示了树的各个级别的命令、带点的名称及相应的输出。

表 21-33 顶层

命令	带点的名称	输出

`list -m`	`server`	`server.applications`
		`server.thread-pools`
		`server.resources`
		`server.http-service`
		`server.transaction-service`
		`server.orb.connection-managers`
		`server.orb.connection-managers.`
		`orb\.Connections\.Inbound\.`
		`AcceptedConnections`
		`server.jvm`

`list -m`	`server.*`	此节点下的子节点的分层结构。
`get -m`	`server.*`	仅显示一条消息，说明此节点上没有属性。

表 21-34 显示了应用程序级别的命令、带点的名称及相应的输出。

表 21-34 应用程序级别

命令	带点的名称	输出

`list -m`	`server.applications`
	或
	`*applications`
		`appl1`
		`app2`
		`web-module1_war`
		`ejb-module2_jar`
		`...`

`list -m`	`server.applications.*`
	或
	`*applications.*`
		此节点下的子节点的分层结构。

`get -m`	`server.applications.*`
	或
	`*applications.*`
		仅显示一条消息，说明此节点上没有属性。

表 21-35 显示了应用程序级别的独立模块和企业应用程序的命令、带点的名称及相应的输出。

表 21-35 应用程序－企业应用程序和独立模块

命令	带点的名称	输出
list -m	server.applications.app1 或 *app1 **注：仅当已部署了企业应用程序时，此级别才可用。如果部署了独立模块，则此级别不可用。**	ejb-module1_jar web-module2_war ejb-module3_jar web-module3_war ...
list -m	server.applications.app1.* 或 *app1.*	此节点下的子节点的分层结构。
get -m	server.applications.app1.* 或 *app1.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.applications.app1.ejb-module1_jar 或 *ejb-module1_jar 或 server.applications.ejb-module1_jar	bean1 bean2 bean3 ...
list -m	server.applications.app1.ejb-module1_jar 或 *ejb-module1_jar 或 server.applications.ejb-module1_jar	此节点下的子节点的分层结构。
get -m	server.applications.app1.ejb-module1_jar.* 或 *ejb-module1_jar.* 或 server.applications.ejb-module1_jar.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.applications.app1.ejb-module1_jar.bean1 **注：在独立模块中，将不显示包含应用程序名称 （本例中为 *app1*）的节点。**	子节点列表： bean-pool bean-cache bean-method

表 21-35 应用程序一企业应用程序和独立模块 (续)

命令	带点的名称	输出
list -m	server.applications.app1.ejb-module1_jar.bean1 **注：在独立模块中，将不显示包含应用程序名称 （本例中为 *app1*）的节点。**	子节点的分层结构及该节点和所有后续子节点的所有属性的列表。
get -m	server.applications.app1.ejb-module1_jar.bean1.* **注：在独立模块中，将不显示包含应用程序名称 （本例中为 *app1*）的节点。**	以下属性及其关联值： CreateCount_Count CreateCount_Description CreateCount_LastSampleTime CreateCount_Name CreateCount_StartTime CreateCount_Unit MethodReadyCount_Current MethodReadyCount_Description MethodReadyCount_HighWaterMark MethodReadyCount_LastSampleTime MethodReadyCount_LowWaterMark MethodReadyCount_Name MethodReadyCount_StartTime MethodReadyCount_Unit RemoveCount_Count RemoveCount_Description RemoveCount_LastSampleTime RemoveCount_Name RemoveCount_StartTime Attribute RemoveCount_Unit
list -m	server.applications.app1.ejb-module1_jar.bean1.bean-pool **注：在独立模块中，将不显示包含应用程序名称 （本例中为 *app1*）的节点。**	没有属性，但显示一条消息，说明“使用带有 --monitor 选项的 get 命令可以查看该节点的属性和值”。
get -m	server.applications.app1.ejb-module1_jar.bean1.bean-pool.* **注：在独立模块中，将不显示包含应用程序名称 （本例中为 *app1*）的节点。**	与 EJB 池属性对应的属性和值的列表 （如表 1-4 所示）。
list -m	server.applications.app1.ejb-module1_jar.bean1.bean-cache **注：在独立模块中，将不显示包含应用程序名称 （本例中为 *app1*）的节点。**	没有属性，但显示一条消息，说明“使用带有 --monitor 选项的 get 命令可以查看该节点的属性和值”。
get -m	server.applications.app1.ejb-module1_jar.bean1.bean-cache.* **注：在独立模块中，将不显示包含应用程序名称 （本例中为 *app1*）的节点。**	与 EJB 高速缓存属性对应的属性和值的列表 （如表 1-5 所示）。

表 21-35 应用程序一企业应用程序和独立模块 (续)

命令	带点的名称	输出
list -m	server.applications.app1.ejb-module1_jar.bean1 .bean-method.method1 **注：在独立模块中，将不显示包含应用程序名称 （本例中为 *app1*）的节点。**	没有属性，但显示一条消息，说明“使用带有 --monitor 选项的 get 命令可以查看该节点的属性和值”。
get -m	server.applications.app1.ejb-module1_jar.bean1 .bean-method.method1.* **注：在独立模块中，将不显示包含应用程序名称 （本例中为 *app1*）的节点。**	与 EJB 模块属性对应的属性和值的列表 （如表 1-2 所示）。
list -m	server.applications.app1.web-module1_war	显示分配给模块的虚拟服务器。
get -m	server.applications.app1.web-module1_war.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.applications.app1.web-module1_war. virtual_server	显示已登记的 servlet 的列表。
get -m	server.applications.app1.web-module1_war. virtual_server.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.applications.app1.web-module1_war. virtual_server.servlet1	没有属性，但显示一条消息，说明“使用带有 --monitor 选项的 get 命令可以查看该节点的属性和值”。
get -m	server.applications.app1.web-module1_war. virtual_server.servlet1.*	与 Web 容器 (Servlet) 属性对应的属性和值的列表 （如表 1-7 所示）。

表 21-36 显示了 HTTP 服务级别的命令、带点的名称及相应的输出。

表 21-36 HTTP 服务级别

命令	带点的名称	输出
list -m	server.http-service	虚拟服务器列表。
get -m	server.http-service.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.http-service.server	HTTP 侦听器列表。
get -m	server.http-service.server.*	仅显示一条消息，说明此节点上没有属性。
list -m	server.http-service.server. http-listener1	没有属性，但显示一条消息，说明“使用带有 --monitor 选项的 get 命令可以查看该节点的属性和值”。
get -m	server.http-service.server.*	与 HTTP 服务属性对应的属性和值的列表 （如表 1-9 所示）。

表 21-37 显示了线程池级别的命令、带点的名称及相应的输出。

表 21-37 线程池级别

命令	带点的名称	输出
`list -m`	`server.thread-pools`	线程池名称列表。
`get -m`	`server.thread-pools.*`	仅显示一条消息，说明此节点上没有属性。
`list -m`	`server.thread-pools.orb\.threadpool\ .thread-pool-1`	没有属性，但显示一条消息，说明“使用带有 --monitor 选项的 get 命令可以查看该节点的属性和值”。
`get -m`	`server.thread-pools..orb\.threadpool\ .thread-pool-1.*`	与线程池属性对应的属性和值的列表（如表 1-14 所示）。

表 21-38 显示了资源级别的命令、带点的名称及相应的输出。

表 21-38 资源级别

命令	带点的名称	输出
`list -m`	`server.resources`	池名称列表。
`get -m`	`server.resources.*`	仅显示一条消息，说明此节点上没有属性。
`list -m`	`server.resources.jdbc-connection-pool -pool.connection-pool1`	没有属性，但显示一条消息，说明“使用带有 --monitor 选项的 get 命令可以查看该节点的属性和值”。
`get -m`	`server.resources.jdbc-connection-pool -pool.connection-pool1.*`	与连接池属性对应的属性和值的列表（如表 1-10 所示）。

表 21-39 显示了事务服务级别的命令、带点的名称及相应的输出。

表 21-39 事务服务级别

命令	带点的名称	输出
`list -m`	`server.transaction-service`	没有属性，但显示一条消息，说明“使用带有 --monitor 选项的 get 命令可以查看该节点的属性和值”。
`get -m`	`server.transaction-service.*`	与事务服务属性对应的属性和值的列表 （如表 1-15 所示）。

表 21-40 显示了 ORB 级别的命令、带点的名称及相应的输出。

表 21-40 ORB 级别

命令	带点的名称	输出
`list -m`	`server.orb`	`server-orb.connection-managers`
`get -m`	`server.orb.*`	仅显示一条消息，说明此节点上没有属性。
`list -m`	`server.orb.connection-managers`	ORB 连接管理器的名称。
`get -m`	`server.orb.connection-managers.*`	仅显示一条消息，说明此节点上没有属性。
`list -m`	`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections`	没有属性，但显示一条消息，说明“使用带有 --monitor 选项的 get 命令可以查看该节点的属性和值”。
`get -m`	`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections.*`	与 ORB 连接管理器属性对应的属性和值的列表 （如表 1-13 所示）。

表 21-34 显示了 JVM 级别的命令、带点的名称及相应的输出。

表 21-41 JVM 级别

命令	带点的名称	输出
`list -m`	`server.jvm`	没有属性，但显示一条消息，说明“使用带有 --monitor 选项的 get 命令可以查看该节点的属性和值”。
`get -m`	`server.jvm.*`	与 JVM 属性对应的属性和值的列表（如表 1-16 所示）。

使用 JConsole

要使 JConsole 能够与 Application Server 一起使用，必须禁用 JMX 连接器的安全性。默认情况下，当前版本的 Application Server （SE/EE 版）已启用安全性。

要禁用 JMX 连接器的安全性，请使用以下方法之一：

1. 使用管理控制台禁用 JMX 连接器的安全性。要从管理控制台执行此操作，请执行以下步骤：
   a. 展开 “配置”节点。
   b. 展开 “管理服务”节点。
   c. 选择 `system` 节点。
   d. 在 "SSL" 部分取消选中 "SSL3" 和 "TLS"。
   e. 选择 “保存”。
2. 使用 `asadmin` 禁用 JMX 连接器的安全性。要从终端窗口或命令提示符执行此操作，请执行以下步骤：
   a. 输入以下命令：

   ```
   asadmin set
   server.admin-service.jmx-connector.system.security-enabled=false
   ```

   b. 重新启动域应用程序服务器 (DAS)。

对于 PE 版， JMX 连接器在默认情况下处于禁用状态，因此无需更改 PE 的任何配置。

3. 启动 JConsole，然后在 “高级”选项卡中输入 JMX URL、用户名和密码进行登录。 JMX URL 的形式为：

   ```
   service:jmx:rmi:///jndi/rmi://< 您的计算机名 >:< 端口
   >/management/rmi-jmx-connector
   ```

   注：如果搜索 `message ADM1501`，您可以从管理 `server.log` 文件中获得确切的 JMX URL。

第 22 章

Java 虚拟机和高级设置

本章介绍了如何配置 Java 虚拟机 (JVM™) 和其他的高级设置。它包含以下各节：

- 用于 JVM™ 设置的管理控制台任务
- 用于高级设置的管理控制台任务

用于 JVM™ 设置的管理控制台任务

- 配置 JVM 常规设置
- 配置 JVM 类路径设置
- 配置 JVM 选项
- 禁用安全管理器
- 配置 JVM 事件探查器设置

配置 JVM 常规设置

Java 虚拟机 (JVM) 包含在 Java 2 Standard Edition (J2SE™) 软件中，它是 Application Server 所需的软件。由于不正确的 JVM 设置将导致服务器停止运行，因此您应当谨慎地更改这些设置。

配置 Application Server 所使用的 JVM 的常规设置：

1. 在树组件中，选择 "Application Server" 节点。
2. 单击 “JVM 设置”选项卡。
3. 默认情况下，选项卡下的 “常规”链接已处于选取状态。
4. 在 “JVM 常规设置”页面中，您可以指定以下内容：
   a. 在 “Java 主目录”字段中，输入 Java 2 Standard Edition (J2SE) 软件的安装目录名称。

      Application Server 依赖于 J2SE 软件运行。要验证此版本是否支持您所指定的 J2SE 版本，请参阅发行说明。（请参见更多信息一节中的链接。）

      注：如果输入不存在的目录名称或输入不受支持的 J2SE 软件版本的安装目录名称，则 Application Server 将无法启动。

   b. 在 "Javac" 字段中，键入 Java 编程语言编译程序的命令行选项。

      当部署 EJB 组件后， Application Server 将运行编译程序。

   c. 要通过 JPDA （Java Platform Debugger Architecture）来设置调试，请选中 “启用调试”复选框并在 “调试选项”字段中指定选项。

      JPDA 供应用程序开发者使用。有关更多信息，请参见 《Application Server Developer's Guide》的 "Debugging J2EE Applications" 一章。（有关指向此指南的链接，请参见 “更多信息”。）

   d. 在 “RMI 编译选项”字段中，键入 RMIC 编译程序的命令行选项。

      当部署 EJB 组件后， Application Server 将运行 RMIC 编译程序。

   e. 在 “字节码预处理程序”字段中，键入以逗号分隔的类名列表。

      每个类都必须实现 `com.sun.appserv.BytecodePreprocessor` 接口。将按指定顺序调用这些类。

      您也许需要在 “字节码预处理程序”字段中输入某些工具 （例如，事件探查器）。事件探查器生成用于分析服务器性能的信息。有关事件探查的更多信息，请参见 《Application Server Developer's Guide》的 "Debugging J2EE Applications" 一章。
5. 单击 “保存”。
6. 重新启动服务器。

配置 JVM 类路径设置

类路径就是 JAR 文件的列表， Java 运行时环境将在此列表中搜索类和其他资源文件。

要配置 Application Server 的 JVM 类路径，请执行以下步骤：

1. 在树组件中，选择 "Application Server" 节点。
2. 单击 “JVM 设置”选项卡。
3. 选择选项卡下面的 “路径设置”链接。
4. 在 “JVM 类路径设置”页面中，您可以指定以下内容：
   a. 在 “环境类路径”复选框中，保留默认选择以忽略 CLASSPATH 环境变量。

      对于编程中的基本教程而言， CLASSPATH 环境变量使用起来很方便，但是不建议将此变量用于企业环境。

   b. 要查看 Application Server 的类路径，请检查 “服务器类路径”字段中的只读内容。
   c. 要将一个 JAR 文件插入到服务器类路径的开头，请在 “类路径前缀”字段中输入此文件的完整路径名。
   d. 要将一个 JAR 文件添加到服务器类路径的末尾，请在 “类路径后缀”字段中输入此文件的完整路径名。

      例如，假设您要指定数据库驱动程序的 JAR 文件。请参见集成 JDBC 驱动程序。

   e. 在 “本地库路径前缀”和 “本地库路径后缀”字段中，您可以将条目添加到本地库路径的开头或末尾。

      将服务器的本地共享库的相对路径、标准 JRE 本地库路径、 shell 环境设置（UNIX 中的 LD_LIBRARY_PATH）以及在 “JVM 事件探查器设置”页面中指定的任意路径顺序连接起来就构成了本地库路径。
5. 单击 “保存”。
6. 重新启动服务器。

配置 JVM 选项

在 “JVM 选项”页面中，您可以为运行 Application Server 的 Java 程序启动（java 工具）指定选项。 -D 选项用于指定 Application Server 的专用属性。

要配置 JVM 选项，请执行以下步骤：

1. 在树组件中，选择 "Application Server" 节点。
2. 单击 “JVM 设置”选项卡。
3. 选择选项卡下面的 “JVM 选项”链接。
4. 在 “JVM 选项”页面中，通过编辑 “值”字段来修改选项。
5. 要添加选项，请执行以下步骤：
   a. 单击 “添加 JVM 选项”。
   b. 在显示的空白行中，在 “值”字段中键入信息。
6. 要删除选项，请执行以下步骤：
   a. 选中该选项旁边的复选框。
   b. 单击 “删除”。
7. 单击 “保存”。
8. 重新启动服务器。

有关 JVM 选项的更多信息，请参见：

- http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html
- http://java.sun.com/docs/hotspot/VMOptions.html

禁用安全管理器

禁用 Application Server 的安全管理器可以提高某些应用程序的性能。即使安全管理器已被禁用， J2EE 授权和验证功能仍可以起作用。您可以在开发环境中禁用安全管理器，但不应在生产环境中禁用安全管理器。

要禁用安全管理器，请执行以下步骤：

1. 转至管理控制台的 “JVM 选项”页面。

   有关说明请参见 “配置 JVM 选项”。

2. 在 “JVM 选项”页面中，删除以下选项：

   `-Djava.security.policy`

3. 单击 “保存”。
4. 重新启动服务器。

配置 JVM 事件探查器设置

事件探查器工具可生成用于分析性能和标识潜在瓶颈的数据。

要为 Application Server 配置事件探查器设置，请执行以下步骤：

1. 在树组件中，选择 "Application Server" 节点。
2. 单击 “JVM 设置”选项卡。
3. 选择选项卡下面的 “事件探查器”链接。
4. 您在 “JVM 事件探查器设置”页面中指定的信息取决于您所使用的事件探查器产品。

   有关示例和说明，请参见 《Application Server Developer's Guide》的 "Debugging J2EE Applications" 一章。（有关指向此指南的链接，请参见 “更多信息”。）

5. 单击 “保存”。
6. 重新启动服务器。

用于高级设置的管理控制台任务

设置高级域属性

1. 在树组件中，选择 "Application Server" 节点。
2. 选择 “高级”选项卡。
3. 在 “域属性”页面中，您可以执行以下操作：
   a. 在 “应用程序根目录”字段中，标识将在其中部署应用程序的完整目录路径。
   b. 在 “日志根目录”字段中，指定保存服务器实例日志文件的目录。
   c. 通常，您需要将 “语言环境”字段保留为空，以使用主机的默认语言环境。

      语言环境是一个标识符，用于指定特定的语言和区域组合。例如，美国英语的语言环境是 en_US，日语的语言环境是 ja_JP。要使用非英语语言环境， Application Server 必须经过本地化，即必须将英语翻译成其他语言。
4. 单击 “保存”。
5. 重新启动服务器。

编译和配置 Apache Web Server

本附录介绍了如何编译 Apache 源代码和配置 Apache Web Server 的安装以使用 Sun Java System Application Server 负载平衡器插件。

请下载相应的 Apache 源代码。有关 Sun Java System Application Server 支持的 Apache Web Server 的版本和平台的信息，请参见 Sun Java System Application Server 发行说明。

本附录包括以下主题：

最低要求

本节介绍了要成功编译 Apache Web Server 以运行负载平衡器插件应满足的最低要求。必须编译和构建 Apache 源代码以使用 SSL 运行。

本节包括以下主题：

适用于 Apache 1.3 的最低要求

有关 Microsoft Windows 平台的要求，请参见：
http://httpd.apache.org/docs/windows.html#req
http://httpd.apache.org/docs/win_compiling.html

适用于其他平台的要求：

- openssl-0.9.7d （源代码）
- mod_ssl-2.8.16-1.3.29 （源代码）
- apache_1.3.29 （源代码）
- gcc-3.3-sol9-sparc-local 软件包 （用于 Solaris 9 SPARC/x86）。
- flex-2.5.4a-sol9-sparc-local 软件包 （用于 Solaris 9 SPARC）
- flex-2.5.4a-sol9-intel-local 软件包 （用于 Solaris 9 x86）

此外，在编译 Apache 之前，请执行以下步骤：

- 在 Linux 上，在同一台计算机上安装 Sun Java System Application Server。
- 在 Solaris 8 上，确保 gcc 和 make 位于 PATH 中。
- 在 Solaris 9 上，确保 gcc 版本 3.3 和 make 位于 PATH 中，并且已安装 flex。
- 如果您在 Red Hat Enterprise Linux Advanced Server 2.1 上使用 gcc，则 gcc 的版本必须在 3.0 以上。

注

- 要使用其他 C 编译器，请设置 C 编译器的路径并使实用程序位于 PATH 环境变量中。例如：

  exportLD_LIBRARY_PATH=$LD_LIBRARY_PATH:appserver_installdir/lib

- 这些软件源可以从 http://www.sunfreeware.com 获得

适用于 Apache 2 的最低要求

有关 Microsoft Windows 平台的要求，请参见：

http://httpd.apache.org/docs-2.0/platform/windows.html

适用于其他平台的要求：

- `openssl-0.9.7d` （源代码）
- `httpd-2.0.49` （源代码）
- `gcc-3.3-sol9-sparc-local` 软件包 （用于 Solaris 9 SPARC/x86）。
- `flex-2.5.4a-sol9-sparc-local` 软件包 （用于 Solaris 9 SPARC）
- `flex-2.5.4a-sol9-intel-local` 软件包 （用于 Solaris 9 x86）

此外，在编译 Apache 之前，请执行以下步骤：

- 在 Linux 上，在同一台计算机上安装 Sun Java System Application Server。
- 在 Solaris 8 上，确保 `gcc` 和 `make` 位于 `PATH` 中。
- 在 Solaris 9 上，确保 `gcc` 版本 3.3 和 `make` 位于 `PATH` 中，并且已安装 `flex`。
- 如果您在 Red Hat Enterprise Linux Advanced Server 2.1 上使用 `gcc`，则 gcc 的版本必须在 3.0 以上。

注

- 要使用其他 C 编译器，请设置 C 编译器的路径并使实用程序位于 PATH 环境变量中。例如：

  ```
  export LD_LIBRARY_PATH=app_server_install_dir/lib:$LD_LIBRARY_PATH
  ```

  该示例是针对 `sh` 的。
- 这些软件源可以从 `http://www.sunfreeware.com` 获得

安装 SSL 可识别 Apache

有关在 Microsoft Windows 平台上编译和安装 Apache 的说明，请参见以下 Web 站点：

Apache 1.3：

http://httpd.apache.org/docs/win_compiling.html

Apache 2：

http://httpd.apache.org/docs-2.0/platform/win_compiling.html

请按照以下步骤在其他平台上编译、配置和安装 SSL 可识别 Apache Web Server。尽管示例显示的是编译和构建 Apache 1.3.29，但相同的过程适用于 Apache 2。

注 在相同的目录级别下对 `mod_ssl`、 OpenSSL 和 Apache 进行解包。

编译和构建 OpenSSL

如果随 Linux 安装的 OpenSSL 的版本为 0.9.7d，则在 Linux 上不需要执行此步骤。

有关 OpenSSL 的更多信息，请参见：

http://www.openssl.org/

解压缩 `openssl-0.9.7d` 源代码并按照以下步骤进行操作。

1. `cd openssl-0.9.7d`
2. `./config`
3. `make`

4. `make test`
5. `make install`

有关通过源代码构建 OpenSSL 的更多信息，请参见 `openssl` 目录中的 `INSTALL` 文件。

使用 mod_ssl 配置 Apache

本节内容仅适用于 Apache 1.3。有关 Apache 2.0 的安装说明，请跳至第 382 页上的“编译和构建 Apache”。

有关 mod_ssl 的更多信息，请参见：

http://www.modssl.org/

1. 下载 `apache_1.3.29` 源代码分发。

   将源代码分发解压缩。源代码分发以压缩归档文件的形式提供。对于 `apache_1.3.29`，源代码分发归档文件将读取 `apache_1.3.29.tar.gz`。

2. 使用以下命令解压缩归档文件：

   ```
   tar -zxvf apache_1.3.29.tar.gz
   ```

   此命令将在当前工作目录中创建名为 `apache_1.3.29` 的目录。

3. 解压缩 `mod_ssl-2.8.14-1.3.29` 源代码。
4. `cd mod_ssl-2.8.14-1.3.29`
5. 运行 `./configure --with-apache=../apache_1.3.29 --with-ssl=../openssl-0.9.7d --prefix=`*install path* `--enable-module=ssl --enable-shared=ssl --enable-rule=SHARED_CORE --enable-module=so`

以上命令示例中指定的目录为变量。 *prefix* 参数指示要将 Apache 安装到的位置。此命令将在屏幕上输出多行。

根据您的系统配置，此命令将为构建创建 `make` 文件。 `configure` 中的错误会导致某些头文件或实用程序丢失。请先安装它们，然后再继续进行操作。

编译和构建 Apache

Apache 的版本不同，适用于编译和构建 Apache 的说明也有所不同。

- 编译和构建 Apache 1.3
- 编译和构建 Apache 2

编译和构建 Apache 1.3

此过程将在第 381 页上的 “使用 mod_ssl 配置 Apache”中介绍的 `--prefix` 属性提供的位置安装 Apache。

1. 在 Linux 上，在 `src/MakeFile` 的 `End of automatically generated section` 之后加入以下行：

   ```
   LIBS+= -licuuc -licui18n -lnspr4 -lpthread -lxerces-c -lsupport
   -lnsprwrap -lns-httpd40

   LDFLAGS+= -L/appserver_installdir/lib
   ```

2. 在 Linux 上，将 Application Server 的安装目录置于 LD_LIBRARY_PATH 中：

   ```
   export LD_LIBRARY_PATH=/app_server_install_dir/lib:$LD_LIBRARY_PATH
   ```

3. 使用 `make` 命令编译 Apache，如下所示：

   a. 使用 cd 命令转至 `mod_ssl` 目录。

   b. `make`

   c. `make certificate`

   d. `make install`

   注 命令 `make certificate` 需要安全密码。请记住此密码，因为启动安全 Apache 时需要它。

   命令 `make install` 将在屏幕上输出多行，表明进程正在编译 Apache 源代码并正在链接 Apache。此进程通常不会出现错误。但如果出现错误，请检查是否已正确下载 Apache 的所有库文件和实用程序。

   通过在 *apache_install_path*`/conf/httpd.conf` 文件中为环境输入适当的值来配置 Apache 的安装。

编译和构建 Apache 2

1. 下载 Apache 2_0_*NN* 源代码分发。

   NN 表示次要版本号，例如， 52。

2. 将源代码分发解压缩。

   源代码分发以压缩归档文件的形式提供。对于 Apache 2_0_*NN*，源代码分发归档文件为 `httpd-2_0_NN.tar.gz`。

3. 使用以下命令解压缩归档文件：

   ```
   tar -zxvf httpd-2_0_NN.tar.gz
   ```

   该命令将在当前工作目录中创建名为 `httpd-2_0_NN` 的目录。

4. 运行以下命令：`cd httpd-2_0_NN`。

5. 运行 `./configure --with-ssl=`*open_ssl_install_path* `--prefix=`*install_path* `--enable-ssl --enable-so`

6. 在 Linux 上，修改 *apache_src*`/build/config_vars.mk` 并添加以下行：

   ```
   EXTRA_LIBS += -licuuc -licui18n -lnspr4 -lpthread -lxerces-c
   -lsupport -lnsprwrap -lns-httpd40

   LDFLAGS+=-L<appserver install dir>/lib
   ```

7. 在 Linux 上，将 Application Server 的安装目录放到 LD_LIBRARY_PATH 中：

   ```
   export LD_LIBRARY_PATH=/app_server_install_dir/lib:$LD_LIBRARY_PATH
   ```

8. 使用 make 命令编译 Apache，如下所示：

   在 `httpd-2_0_NN` 目录中执行以下命令：

   a. `make`

   b. `make install`

   命令 `make install` 将在屏幕上输出多行，表明进程正在编译 Apache 源代码并正在链接 Apache。此进程通常不会出现错误。但如果出现错误，请检查是否已正确下载 Apache 的所有库文件和实用程序。

   通过在 *apache_install_path*`/conf/httpd.conf` 文件中为环境输入适当的值来配置 Apache 的安装。

注 如果遇到错误，请尝试将 Application Server 安装目录置于 PATH 中：

```
export LD_LIBRARY_PATH=/app_server_install_dir/lib
```

或者添加 OpenSSL 库，例如：

```
export
LD_LIBRARY_PATH=/openssl_install_dir/lib:/app_server_install_dir/lib
```

注 在 Apache 2 上，您必须手动创建并安装证书。有关更多信息，请参见 Apache 文档。

启动和停止 Apache

Apache 附带了名为 `apachectl` 的脚本，此脚本可简化 Apache 的启动、停止和重新启动。

- 运行以下命令以启动 Apache：

  apache_install_dir`/bin/apachectl start`

- 运行以下命令以在 SSL 模式下启动 Apache：

  apache_install_dir`/bin/apachectl startssl`

- 要停止 Apache，请运行以下命令：

  apache_install_dir`/bin/apachectl stop`

启动 Apache 之后，请测试安装。运行 Apache 后，请在 Web 浏览器中键入以下地址：http://*server_name*:*port_number*/. 如果安装成功且 Apache 正在运行，将显示测试页面。

完成 Apache 安装后，请参见第 66 页上的 “对 Apache Web Server 的修改”以获得有关在安装插件过程中和安装插件后的 Apache 配置的信息。

自动重新启动域或节点代理

如果域或节点代理被意外停止（例如，您需要重新启动计算机），您可以将系统配置为自动重新启动域或节点代理。

本附录包括以下主题：

在 UNIX 平台上自动重新启动

要在 UNIX 平台上重新启动域，请在 `/etc/inittab` 文件中添加一行文本。

例如，要重新启动安装在 `opt/SUNWappserver` 目录中的 Application Server 的 domain1，请使用名为 `password.txt` 的密码文件。

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user admin
--passwordfile /opt/SUNWappserver/password.txt domain1
```

将这些文本放在一行上。前三个字母是进程的唯一指示符，可以进行更改。

重新启动节点代理的语法与此相似。例如，要重新启动安装在 `opt/SUNWappserver` 目录中的 Application Server 的 agent1，请使用名为 `password.txt` 的密码文件。

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-node-agent --user admin
--passwordfile /opt/SUNWappserver/password.txt agent1
```

在 Microsoft Windows 平台上自动重新启动

要在 Microsoft Windows 上自动重新启动，请创建一个 Windows 服务。结合使用 Sun Java System Application Server 附带的 appservService.exe 和 appserverAgentService.exe 可执行文件以及 Microsoft 提供的 Service Control 命令 (sc.exe)。

Windows XP 附带了 sc.exe 命令，该命令位于 C:\windows\system32 目录或 C:\winnt\system32 目录中。编写本文档时，已经可以从以下地址下载 Windows 2000 sc.exe：ftp://ftp.microsoft.com/reskit/win2000/sc.zip°£ 有关使用 sc.exe 的更多信息，请参见 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndllpro/html/msdn_scmslite.asp。

使用 appservService.exe 和 appservAgentService.exe，如下所示：

```
C:\winnt\system32\sc.exe create service_name binPath=
\"fully_qualified_path_to_appservService.exe \"fully_qualified_path_to_asadmin.bat start_command\"
\"fully_qualified_path_to_asadmin.bat stop_command\"" start= auto DisplayName=
"display_name"
```

例如，要创建用于启动和停止域 domain1 的名为 SunJavaSystemAppServer DOMAIN1 的服务，请使用密码文件 C:\Sun\AppServer\password.txt：

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe \"C:\Sun\AppServer\bin\asadmin.bat
start-domain --user admin --passwordfile C:\Sun\AppServer\password.txt
domain1\" \"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\"" start=
auto DisplayName= "SunJavaSystemAppServer DOMAIN1"
```

要创建用于启动和停止节点代理 agent1 的服务，请使用：

```
C:\windows\system32\sc.exe create agent1 binPath=
"C:\Sun\AppServer\lib\appservAgentService.exe
\"C:\Sun\AppServer\bin\asadmin.bat start-node-agent --user admin
--passwordfile C:\Sun\AppServer\password.txt agent1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-node-agent agent1\"" start= auto
DisplayName= "SJESAS_SE8.1 AGENT1"
```

注 作为 binPath= 参数的一部分而输入的启动和停止命令必须具有正确的语法。要进行测试，请在命令提示符下运行这些命令。如果这些命令不能正确地启动或停止域或节点代理，则说明该服务不能正常工作。

注 请勿混合使用 asadmin 启动和停止命令与服务启动和停止命令。否则，可能导致服务器状态不同步。例如，即使组件没有运行，服务可能也不会显示已启动组件。为避免发生这种情况，在使用服务时请始终使用 sc.exe 命令来启动和停止组件。

自动重新启动的安全性

通过以下方式之一来处理启动时所需的密码和主密码：

- 在 Microsoft Windows 上，将服务配置为要求用户输入密码。
  a. 在 “服务”控制面板中，双击创建的服务。
  b. 在 “属性”窗口中，单击 “登录”选项卡。
  c. 选中 “允许服务与桌面交互”，以便在启动组件时提示输入所需的密码。

     必须登录才能看到提示，键入条目时条目不会在屏幕上显示出来。该方法是使用服务选项的最安全的方法，但在服务可用之前需要进行用户交互。

     如果未设置 “与桌面交互”选项，则服务将处于 “启动 - 暂挂”状态并显示为挂起。中止服务进程即可从此状态中恢复。
- 在 Windows 或 UNIX 上，使用 --savemasterpassword=true 选项创建一个域，并创建一个存储管理员密码的密码文件。启动组件时，使用 --passwordfile 选项来指向包含密码的文件。

  例如：

  a. 创建具有已保存的主密码的域。在下面的语法中，系统将提示您输入管理员密码和主密码：

     ```
     asadmin create-domain --adminport 4848 --adminuser admin
     --savemasterpassword=true --instanceport 8080 domain1
     ```

b. 在 Windows 上，创建使用密码文件的服务，以输入管理员密码：

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe
\"C:\Sun\AppServer\bin\asadmin.bat start-domain --user admin
--passwordfile C:\Sun\AppServer\password.txt domain1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\"" start=
auto DisplayName= "SJESAS_PE8.1 DOMAIN1"
```

密码文件 `password.txt` 的路径为 `C:\Sun\AppServer\password.txt`。该文件包含以下格式的密码

`AS_ADMIN_password=`*password*

例如，密码 `adminadmin` 在该文件中的格式为：

```
AS_ADMIN_password=adminadmin
```

c. 在 UNIX 上，请使用添加到 `inittab` 文件的行中的 `--passwordfile` 选项：

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user
admin --passwordfile /opt/SUNWappserver/password.txt domain1
```

密码文件 `password.txt` 的路径为 `/opt/SUNWappserver/password.txt`。该文件包含以下格式的密码

`AS_ADMIN_password=`*password*

例如，密码 `adminadmin` 在该文件中的格式为：

```
AS_ADMIN_password=adminadmin
```

domain.xml 的带点名称属性

本附录介绍了可用于描述 Mbean 及其属性的带点名称属性。 `domain.xml` 文件中的每个元素都有相应的 MBean。由于用于使用这些名称的语法涉及使用句点来分隔名称，因此这些名称被称为 “带点名称”。

本附录包括以下主题：

顶层元素

domain.xml 文件中的所有顶层元素都必须满足以下条件：

1. 每个服务器、配置、群集或节点代理的名称都必须是唯一的。
2. 不能将服务器、配置、群集或节点代理命名为 "domain"。
3. 不能将服务器实例命名为 "agent"。

下表列出了顶层元素及其相应的带点名称前缀。

表 C-1 顶层元素

元素名	带点名称前缀
applications	domain.applications
resources	domain.resources
configurations	domain.configs
servers	domain.servers
	该元素包含的所有服务器都可以作为 *server-name* 被访问。其中，*server-name* 是服务器子元素的名称属性值。
clusters	domain.clusters
	该元素包含的所有群集都可以作为 *cluster-name* 被访问。其中，*cluster-name* 是群集子元素的名称属性值。
node-agents	domain.note-agents
lb-configs	domain.lb-configs
system-property	domain.system-property

有两个可用的别名级别：

1. 利用第一级别的别名可以访问服务器实例或群集的属性而不必通过 domain.servers 或 domain.clusters 前缀。因此，举例来说，形式为 "server1" 的带点名称将映射到带点名称 domain.servers.server1 （其中 server1 为服务器实例）。
2. 第二级别的别名用于表示群集或独立服务器实例 （目标）的配置、应用程序和资源。

下表列出了以服务器名称或群集名称开头的带点名称，这些带点名称被别名化为域下的顶层名称：

表 C-2 域下的带点名称服务器名称

带点的名称	别名化为	说明
target.applications.*	domain.applications.*	该别名将解析为仅由目标引用的应用程序。
target.resources.*	domain.resources.*	该别名将解析为由目标引用的所有 `jdbc-connection-pool`、`connector-connection-pool`、`resource-adapter-config` 和所有其他资源。

下表列出了以服务器名称或群集名称开头的带点名称，这些带点名称在服务器或群集所引用的配置中被别名化的顶层名称。

表 C-3 服务器或群集引用的配置的带点名称

带点的名称	别名化为
target.http-service	*config-name*.http-service
target.iiop-service	*config-name*.iiop-service
target.admin-service	*config-name*.admin-service
target.web-container	*config-name*.web-container
target.ejb-container	*config-name*.ejb-container
target.mdb-container	*config-name*.mdb-container
target.jms-service	*config-name*.jms-service
target.log-service	*config-name*.log-service
target.security-service	*config-name*.security-service
target.transaction-service	*config-name*.transaction-service
target.monitoring-service	*config-name*.monitoring-service
target.java-config	*config-name*.java-config
target.availability-service	*config-name*.availability-service
target.thread-pools	*config-name*.thread-pools

不能别名化的元素

不应对群集实例进行别名化。要获得群集实例的系统属性，应按以下方法使用带点名称属性：domain.servers.clustered-instance-name.system-property，而不是clustered-instance-name.system-property。

索引

英文

A

B

C

D

F

G

H

J

K

L

M

N

P

Q

R

S

T

W

X

Y

Z

Z